ECHO Users' Guide

for ECHO Participants

ECHO Users Guide

To accompany ECHO Version 7.0
February 2006

Contents

Figures

Tables

Code Examples

Error Examples

ECHO User's Guide

Preface

The Earth Observing System (EOS) Clearing HOuse (ECHO) system became operational in November 2002 with Version 4.5 (Operational Release 1). This document describes the Version 6.0 operational release. The ECHO system has evolved from a prototype to an operational system, and as such will continue to evolve as user concerns are addressed. The evolution should be primarily in the addition of new features, but may also involve changes to key parts of the system. These changes are largely derived from the data model review, and have been approved for the system. The ECHO project will make every effort to summarize these changes so that developers will have the opportunity to adapt quickly. Furthermore, the ECHO Development Team will strive to incorporate them as quickly as possible, so that partners are impacted as little as possible.

This document is intended to provide the reader with approaches to using ECHO but not necessarily the specific details of every transaction. The online Application Program Interface (API) guide (http://api.echo.nasa.gov/echo/message_detail.html) provides the details of how to create each message, but does not provide the big picture.

Conventions

All references to time are in Greenwich Mean Time (GMT).

1 Introduction to ECHO

ECHO functions as a metadata clearinghouse and order broker for Earth Science Data and Information System (ESDIS) data and services applied to that data. ECHO hosts a cache of metadata representing the data holdings of a wide variety of partners. It adds value to existing systems by providing a single portal on the Internet where these metadata can be searched.

1.1 ECHO Concept and Design

1.1.1 System Description

ECHO's framework of components and a spatially enabled database function together as a clearinghouse and order broker for Earth Science metadata. ECHO also incorporates a Universal Description, Discovery, and Integration (UDDI) registry that facilitates registration, discovery, and invocation of services related to the ECHO holdings. Internally, ECHO specifies application process interfaces (APIs) and provides middleware components (including data and service search and access functions) in a layered architecture. Figure 1 depicts the ECHO system context in relation to its public APIs.

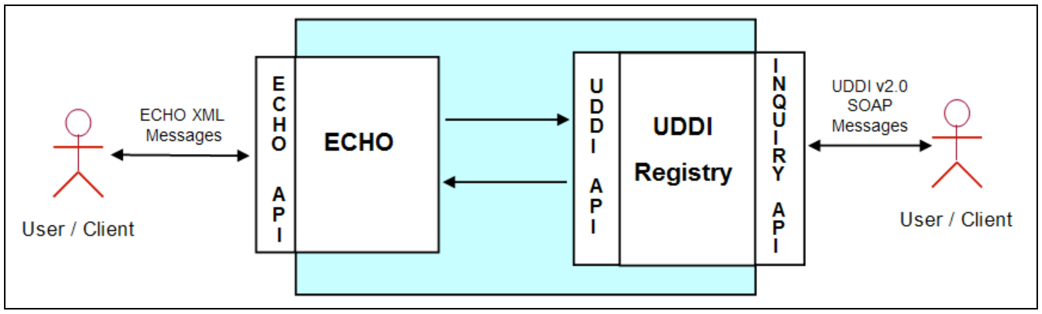


Figure 1. ECHO System Context

ECHO allows partners to cache copies of their metadata within it. Partners have complete control over what metadata is represented in ECHO on their behalf. They can insert new data, modify existing data and remove old data. Clients representing various entities can communicate with ECHO via its APIs in order to perform certain functions on ECHO's holdings.

All metadata is held in an Oracle database with spatial extensions. The metadata model is derived directly from that used by the Earth Observing System Data and Information System (EOSDIS) Core System (ECS). Additional information that ECHO persists is also maintained in the database as persisted objects represented in a relational form.

Key features of the ECHO architecture are ease of partner participation, data model consistency, API specifications that enable direct user/scientist participation, the extensibility of user interface and an evolutionary development approach that can exploit industry innovations.

Ease of Partner Participation: Designed to be low-cost and minimally intrusive, ECHO offers a set of standard ways for partners to interface with the system and a metadata exchange approach that accommodates existing partners and technology.

Data Model Consistency: To mitigate the risk of not being able to match all possible partner data models, ECHO has prototyped a Metadata Mapping Tool to translate non-standard formats upon ingest into ECHO.

Open System/Published APIs: ECHO uses an open system approach and ensures that user interfaces fully address user/scientist needs by specifying and publishing domain APIs that accommodate independent ECHO clients. These APIs are independent of the underlying transport protocols used. Currently, ECHO is capable of communicating using Simple Object Access Protocol (SOAP) and Java Remote Method Invocation (RMI). Plans are in place to add a Web services view of ECHO services. Other transport protocols can be added as necessary. Interactions with ECHO may involve user interaction in real time or may be machine-to-machine.

Extensibility of User Interface: ECHO extensibility is ensured by its component architecture, which allows new capabilities and functions to be plugged in, modeling relationships between services/APIs/UIs, and continued prototyping. ECHO's current focus is on the middleware and on enabling many different types of user interfaces via its APIs.

Evolutionary Development: The ECHO system is developed in increments to allow for insight and feedback during the development cycle. Industry trends are followed and the use of commercial, off-the-shelf products is optimized.

1.1.1.1 ECHO as an API Framework

ECHO was designed to provide a message-passing infrastructure. The assumption is that the API functions by allowing an eXtensible Markup Language (XML) message to be passed to it, and then it responds with an XML message of its own. Messages passed in and received from the ECHO API interface conform to a set of Data Type Definitions (DTDs) that correspond to the particular messages involved. The DTD acts as a template, describing generally how a message should appear, but not every aspect. The DTD, combined with information about how to semantically interpret the message, allows an external entity to create messages to be passed to ECHO and to interpret its results.

ECHO provides functions that are applicable across all API accesses to the system, and are therefore part of the framework. First, the ECHO framework provides an RMI-based interface. This interface defines three basic functions that an ECHO client can perform: login, logout, and perform a function (transaction) that is simply a way to input an XML string and receive an XML string as a result.

Second, ECHO provides a configurable translator system that converts an XML message into the appropriate Java method invocation. This translation improves processing by allowing the internals of ECHO to function in its native language. It also allows the addition, removal and modification of existing messages through a configuration process that does not involve coding, thus providing a more adaptable code base.

Third, ECHO groups transactions into services. Each service limits types of users who are allowed to connect to it. For instance, transactions associated with a provider updating its profile information (address, email, contacts, etc.) are only executable by users who are logged in to the system in a provider role. See sections on Users and Roles. Each service has a fixed set of user types that can use the service.

1.1.1.2 ECHO as a Spatially Enabled Metadata Search and Order System

Having defined a system that understands how to exchange XML messages, the next step is to have a system that understands spatially enabled Earth Science metadata. This is accomplished by introducing Oracle with its spatial extensions into the system and creating business logic within the system that understands how to interact with that metadata. In addition, a second ECHO interface is added, which allows metadata updates to go directly into the database bypassing the message-passing API. A File Transfer Protocol (FTP) server is configured to receive these update files, which are also expressed in XML according to two DTDs, one for granules (or inventory) and one for collections (or datasets).

ECHO uses Oracle's spatial capabilities to search for data whose spatial extent is described within the system. A partner can associate boxes, circles and polygons with a granule or a collection. A client can then construct a search using these types of regions and ECHO responds with data whose spatial region intersects the described region.

ECHO provides services for interacting with this catalog of metadata. Queries can be performed in a number of ways, result formats can be specified (what fields should be returned), and the resulting data sets can be incrementally accessed so that large return sets can be handled gracefully. ECHO also supports constructing, submitting, and tracking orders for the data that the metadata represents. ECHO supports both an embedding of a Uniform Resource Locator (URL) within the metadata for accessing the data (which the client simply accesses via Hypertext Transfer Protocol [HTTP]), or a more complicated order process in which quotes and order options are accommodated.

The query language can be specified and must be embedded within a query transaction as a string. Query language specification allows the system to support multiple query languages. Currently only one is implemented, which is a domain-based query language built on XML and modeled after the V0 Object Description Language (ODL) queries.

1.1.1.3 ECHO as an Earth Science Metadata Search and Order System

The ECS model is used as a basis for ECHO. ECHO incorporates the ECS concept of granules and collections and defines separate DTDs for updating each of these, assuming that granules will indicate which collection they consider their primary collection. A granule is the smallest aggregation of data that is independently managed (described, inventoried, and retrieved). A collection is a grouping of 0 or more granules that all come from the same source, such as a modeling group or institution.

“Primary collection” means the collection that owns the granule – the collection from which to order that granule. Collections contain information common across all the granules they contain, plus a template for describing additional attributes to define data fields in ECHO that encapsulate information not already part of the metadata model. Granules have their own metadata model and support values associated with the additional attributes defined by the owning collection.

1.1.1.4 ECHO as a UDDI Registry

The ECHO Extended Services Registry component is a public UDDI registry deployment. A UDDI registry enables organizations offering services to register and classify those services so that service consumers may discover them. For Web Services, a UDDI registry stores and provides information sufficient for service consumers to directly invoke the services offered by the partner. Figure 2 depicts this “publish, find, & bind” paradigm. The UDDI registry provides a central location for information publication and discovery; however, it does not participate directly in transactions that may occur between consumers and partners.

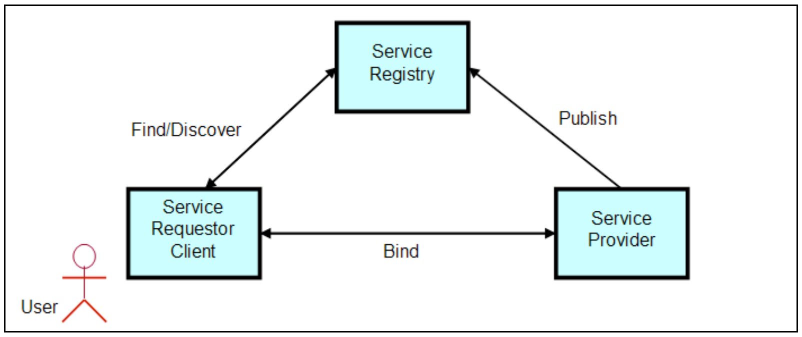


Figure 2. UDDI Publish, Find, and Bind Paradigm

There are standard programming interfaces (APIs) defined for all instances of UDDI registries. The standard UDDI interface comprises two distinct interfaces: the "Inquiry API" and the "Publishing API". The ECHO system makes the Inquiry API, which is used for discovering and retrieving information about registry content, available as a public interface. This interface describes a web service, which receives SOAP-formatted messages. In accordance with the UDDI specification, these messages "all behave synchronously and are required to be exposed via HTTP-POST only." Information about the UDDI Inquiry API can be found at the following location:

http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm#_Toc25137711

1.1.1.5 Security

The ECHO system supports Secure Sockets Layer (SSL)- based communication, which can be used by a client to pass passwords or other sensitive information securely. Internally, the systems are firewalled to prevent unintended access.

1.1.1.6 Supported Platforms

The ECHO system supports clients capable of initiating a Remote Method Invocation (RMI) or SOAP connection.

1.1.2 System Drivers

In order to address effectively the system vision outlined above, ECHO also had to respond to a set of system drivers. These drivers, derived from functional, organizational and operational concerns expressed by the user community, determined the architectural approach and the types of technical solutions used in building the system.

1.1.2.1 Ease of Participation

The primary goal of ECHO is to enable organizations to participate in making their resources and capabilities available to the Science community. In order to facilitate participation by these organizations, ECHO has:

- Minimized the number of requirements that a partner must meet in order to participate.
- Involved partners in the system's development cycle and requirements definition.
- Selected metadata insert and update mechanisms based on current standard industry practice (e.g., XML) that most databases can generate automatically.
- Provided mapping capabilities to convert from one XML representation into another.

1.1.2.2 Cost to Field

While aggressive in the capabilities it is targeted to support, ECHO continually evaluates performance and functionality against costs in order to minimize the Cost to Field, e.g., licensing of COTS, amount of custom code required, hardware platform requirements, and complexity of networking and installation.

1.1.2.3 Cost to Operate

Once fielded, ECHO seeks to minimize the cost to operate the system. Every effort is made to minimize requirements for operations staff, in both skill and quantity.

1.1.2.4 Cost to Maintain

ECHO's architecture, development processes, and use of commercial products have all been selected to minimize the costs involved in maintaining the system, e.g., programming staff required for evolutionary improvements, maintenance costs of COTS products, potential functional enhancements.

1.1.2.5 Extensibility

ECHO is being built with long-term extensibility foremost in mind. In order to enable emerging techniques and strategies for Earth science research, ECHO has:

- Adopted a "design for change" as a goal at the beginning of ECHO development.
- Built in the capability to limit the impact of changes to the API to configuration file, the service interface, and the business logic that actually implements the function.
- Developed test tools to regression test large portions of functionality automatically overnight, so when changes are made, undesirable impacts can be discovered quickly.
- Adopted a layered ECHO architecture to allow changes to one component of the system without affecting other components.

1.1.3 ECHO Capability/Functionality

ECHO provides an infrastructure that allows various communities to share tools, services and metadata. As a metadata clearinghouse, it supports old and new data access paradigms such as navigation and discovery. As an order broker, it forwards orders for data discovered through the metadata search process to the partners for satisfaction. As a service broker, ECHO decentralizes end user functionality and supports interoperability of distributed functions.

ECHO currently supports partners at three different levels: Data Partner, Client Partner and Service Partner. These different levels of participation are not exclusive.

Data Partners – Organizations that supply inventory-level metadata representing their data holdings to ECHO.

Client Partners – Organizations that participate by developing software applications to access the Earth Science metadata.

Service Partners – Organizations that participate by advertising their Earth Science-related services to the user community via ECHO. ECHO will maintain service descriptions in a service catalog (either special services, or services that are available as an option on a selected set of granules/collections) and support the user in ordering those services.

ECHO addresses Science User needs through a set of well-defined and open interfaces upon which the user community can build its own client applications. In this way, ECHO supports extendable, flexible user interfaces, allowing industry and the community to drive the progress of available Earth Science applications.

Groups outside the Earth Science Data and Information System (ESDIS) community can subscribe to metadata holdings in order to build their own systems. The ECHO approach allows users to build their own user interfaces (clients) to ECHO, so they are not limited by the data search and order system provided by NASA. For data partners, ECHO offloads the burden of providing the system resources required for searching, and gives users the flexibility to support community-specific services and functionality. ECHO's interoperability features allow all participants to benefit from the distributed development of functions, again reducing dependence on NASA resources.

1.2 How to Talk to ECHO

One system driver in designing ECHO is that there should be programmatic ways to access the system. In other words, the system should not rely only on a Graphical User Interface (GUI) for user access. All functions of the system can be accessed programmatically. Specifically, a user of the system need not be a person, but could be another computer. This focus has led to the API interface being developed without a user interface. The intent is to allow many varied user interfaces, each of which addresses its own community of users. This section describes how to communicate with ECHO via the APIs. Other methods, such as FTP for metadata ingest, are discussed in subsequent sections.

1.2.1 Message-Based APIs

The majority of all interactions with the system take place through a message-based API. The basis of these messages is XML. XML provides a structure and syntax, but does not specify the semantics of a message. ECHO accepts a message in XML, performs the requested transaction, and then responds with any results and status in XML as well. Currently, messages are passed into ECHO through Java RMI. With the exception of session management (logging in and out), all interactions consist of simply sending an XML string and receiving one in return. Because of the textual nature of this interaction, the protocol used to exchange messages is not critical.

As noted above, the exception to this is session management. It is important to have a method of establishing a session as a known user and then disconnecting from it. In the current ECHO implementation, there are additional RMI method calls for login and logout. If other protocols are used, then some secure method for establishing a session must be created.

API transactions are divided into groups known as services. Each service has a number of transactions grouped according to their actions, and only allows users of certain types to connect to it.

1.2.1.1 The Session Manager

All communication with ECHO takes place through the Session Manager. A client uses a SOAP client library to obtain a reference to ECHO's Session Manager, and then sends all communication to ECHO through that reference. The Session Manager interface is shown in Table 1:

Method Summary		
Return Type	**Message Signature**	**Description**
void	identify(String client)	Used to identify the client accessing ECHO. All clients should call this after establishing a session.
boolean	login(String username, String password)	Used to establish a registered user session.
void	logout()	Used to end a registered user session.
String	perform(String msg)	Access to ECHO's XML message based services.
boolean	setProviderContext(String providerId)	Used by a registered user to specify which provider they are acting as (if they can represent more than one).
String	getECHOVersion()	Used to retrieve the current version of the ECHO API

Table 1: Session Manager Interface.

A client, after obtaining a reference to the Session Manager, should identify itself through the identify method. The string submitted should be a brief identifier for the client and its version number (e.g., WIST v1.3). Once this is done, the perform method can be used to submit XML messages for service requests that match the ECHO API, and the resulting XML message is returned from it. If login has not been called, then the client is acting as a guest user. If login has been called, then the client is acting as that user. It is expected that the client will simply provide a userid and password prompt for its user, and then pass that information to ECHO. Logout can be used to end a registered user's session. SetProviderContext is used when performing provider maintenance functions to identify which provider a registered user is working for, if that registered user has more than one provider role (See roles description in section 2.4). If there is only one provider role for the registered user, then it is not necessary to call this function, as ECHO knows which provider to reference. If there are no provider roles for a registered user, then that user cannot execute the provider management functions.

If it is desired to know the version number of the ECHO api with which a client is communicating, the client can directly invoke the getECHOVersion method through ECHO client proxies.

1.2.1.2 XML Basics

It is important for developers to have a fundamental understanding of ECHO. This section strives to provide basic information needed to get started with ECHO and is not intended to be an XML tutorial.

Note: If one decides to use the Java Façade, then this level of complexity is hidden from the client.

ECHO uses DTDs as the templates for the messages that it handles. These templates can be used to help construct a message the first time, and to validate that a message is constructed correctly. The basic structure of an ECHO API message is:

```
<ServiceName><TransactionName><TransactionSpecificParameter>…</TransactionSpecificPara
meter></TransactionName></ServiceName>
```

Note that the XML structure is similar to Hypertext Markup Language (HTML). Each part begins with an open tag inside of angle brackets, and ends with the same tag with a leading slash. The first level tells what individual service of ECHO is being addressed. The second level identifies the particular transaction that is being requested. If there are no parameters, then that is all that is needed, otherwise the parameters are embedded inside at this level according to the appropriate DTD.

1.2.1.3 XML Escape Characters

Partners must supply a valid XML file for their metadata. A metadata input XML file should be validated against its corresponding DTD before being sent to ECHO. Certain characters such as “&” etc. in the XML file must be addressed. The input string containing those characters can be either expressed using CDATA expression or using escape characters.

The following examples shown how the string should be given in the XML file:

for long name:

```
<name part 1 & name part 2>
```

using CDATA expressions:

```
<LongName>![CDATA[<name part 1 & name part 2>]]</LongName>
```

using escape characters:

```
<LongName>&lt;name part 1 &amp; name part 2&gt;</LongName>
```

The ECHO ingest process handles the escape characters in Table 2: if they appear in the input XML file.

Table 2: XML escape characters for ingest.

Character	Escape Character
&	&amp;
<	&lt;
>	&gt;
'	&apos;
"	&quot;

1.2.2 UDDI Inquiry API

The ECHO UDDI Registry “Inquiry” API accepts SOAP-formatted messages. The following paragraph relates SOAP to UDDI and can be found in the UDDI specification. Note that the ECHO UDDI registry implements the UDDI v2 standard.

“Simple Object Access Protocol (SOAP) is a method for using XML in message and remote procedure call (RPC) based protocols. SOAP has been jointly defined and submitted to the World Wide Web consortium (W3C) as a recommendation. UDDI uses SOAP in conjunction with HTTP to provide a simple mechanism for passing XML messages to Operator Sites using a standard HTTP-POST protocol. Unless specified, all responses will be returned in the normal HTTP response document. As of version 2, there are still no interactions that deviate from this general rule.”

2 ECHO Entities

Interactions with ECHO are done through message passing and affect the state of entities within the system. The major ones are described in this section.

2.1 Users

The most basic entity in the ECHO system is a User. Users can be distinguished from each other based on their username, which is unique to any particular user of the system. There are two types of users: registered users and guests. Guests have the ability to do many of the things registered users can do, but they cannot count on persistent access to information across sessions. Registered users can save information to be used in their next session after they log out of the current one. Registered users can further be segregated into Providers and Clients. Provider users are associated with a data center. Clients are associated with an end client user.

2.2 Roles

2.2.1 User Access

In the past, there was a concrete concept of two kinds of registered users that could log into the system – the science user and the provider user. Now, only one kind of user can log into the system – the science user. All authorization in the system has moved from the service layers to the transactions themselves. Provider-oriented transactions are no longer used by provider users but instead facilitated through the concept of provider roles that are associated with a science user.

ECHO now has a concept of "user roles". User roles are a way to grant a user access to the system. These roles facilitate greater flexibility with transaction-level authorization and allow certain users the ability to execute both client and provider transactions without having two accounts in the system. Currently, there are two kinds of roles:

1. Provider Role: A user can have one or more provider roles, with each role associated with one provider in the system. A user with one or more provider roles can access and update information about the providers with which they are associated. For instance, if a user has a provider role for Oak Ridge National Laboratory (ORNL), then that user can use the UpdateContact transaction in ProviderAccountService to update the contact information for ORNL.

2. Admin Role: A user can only be associated with one instance of this role. This role allows the user to access and update information about any user and any provider. Essentially, this role should allow the user full access to almost anything in the system and available through the current API. Users who are assigned this role should read the ECHO Operations Guide as well as this document.

2.2.2 Transactions

There are several transactions in ECHO to facilitate the use of these roles. They are in ProviderAccountService, AdministrationService, UserAccountService, and in SessionManager.

2.2.2.1 SetProviderContext

If the user is associated with one or more provider roles, there must be a way to tell the system which provider they want to currently represent. Thus, each user has what is called a provider context. The provider context holds the current provider that user represents, and as the name implies, it is in the context of this provider to which the provider-oriented transactions in the system are applied.

Note: If the user is only associated with one provider role, using this transaction is not necessary. The system will assume that the user's 'provider context' holds the provider associated with that one provider role.

2.2.2.2 SetUserContext

This transaction is similar in concept to the 'SetProviderContext' transaction in ProviderAccountService. Specifically, an admin's 'user context' holds the current user to which the admin represents, and as the name implies, it is in the context of this user to which user-oriented transactions in the system are applied.

Note: Unlike the SetProviderContext transaction, there is no case where the system can assume which user the admin should represent. Thus, if an admin user does not set their 'user context' before executing a user-oriented transaction, then the transaction is applied to the actual admin user. Also there is no way to empty the 'user context' after it is set. Thus, if the admin user needs the user-oriented transactions to apply to oneself then the admin user must set the 'user context' to the actual admin user.

2.3 Queries

A Query in the ECHO system is used to search and retrieve science metadata stored by ECHO. A user can create a query by issuing a QueryRequest message of the CatalogService to ECHO. Users can search on collections (referred to as a "Discovery Search," or on granules (referred to as an "Inventory Search"). Table 3: and Table 4: identify the kinds of queries that ECHO supports, with examples of what that search might yield.

Table 3: Discovery Search – Collections.

Search Criteria	Search for collections based on
Campaign	'Soil Collections','River Discharge (RIVDIS)'
Dataset ID	'15 MINUTE STREAM FLOW DATA: USGS (FIFE)'
ECHO Insert Date	The date it was inserted into ECHO; e.g. from July 3, 2001 at 5:30 pm to June 2, 2002
ECHO Last Update	The date the collection level metadata for this collection was last updated in ECHO.
Online Collections Only	Only those that are available online
Parameter	Geo-physical parameter; e.g.
Processing Level	1A, 2B, etc.
Sensor Name	'BAROMETER', 'CAMERA', 'SWIR'
Short Name	'AST_L1A', 'MYDPT1KN'
Source Name	'DIGITAL ELEVATION MODEL', 'NOAA-9'
Spatial	Polygon, polygon with holes, multi polygon
Spatial Keywords	'Global', 'United Kingdom', 'Solar'
Temporal	Date range and periodic ranges search over the temporal coverage of the collection
Temporal Keywords	'January', 'Spring', 'Stone age', 'Weekly'
PSA Names	'DAR_ID', 'QAFRACTIONGOODQUALITY','FIREPIXELS'
Orderable items	Collections available for ordering

Table 4: Inventory Search – Granules.

Search Criteria	Search for collections based on
Only Granules with Browse Data	Granules that have associated browse in ECHO
Campaign	'Soil Collections','BOREAS'
Percentage of Cloud Cover	Percentage of cloud cover between 5-10%
Dataset ID	Granules from the collection identified by dataset id
ECHO Insert Date	Date it was inserted into ECHO
ECHO Last Update	Granule metadata last updated in ECHO
Either Day or Night granules	Whether they are taken in day time, night time or both
Only Global Granules	Whether the granules are global
Search on granule IDs (ECHO specific)	Only search for specific granules using ECHO ID
Search on Granule UR	'SC:AST_L1A.003:2007226177', 'VEMAP_1_SITE.w'
Online Granules Only	Only those that are available online
Two Dimensional Coordinate System	Any two-dimensional coordinate system-based search criteria supported by the collection, such as path/row
Local Granule ID	Search for granule using science team ID
Sensor Name	'BAROMETER', 'CAMERA', 'SWIR'
Source Name	'DIGITAL ELEVATION MODEL', 'NOAA-9'
Spatial	Polygon, polygon with holes, multi polygon
Temporal	Temporal coverage of the granule
PSA	E.g. granule for DAR_ID value '345f3c'
Orderable Items	Granules available for ordering

ECHO supports storage and retrieval of previously stored queries by users. A user can also execute a previously stored query to get data that are more recent.

Currently the QueryRequest message can be used only to perform a synchronous query. This may be a drawback if the query takes too long. In future releases of ECHO, the users will be able to execute a Query asynchronously. In this context, a synchronous query is one in which the query will execute before a response is returned to the query message. An asynchronous query will receive a response to its initiating message immediately and then require an additional message to find out that the query has completed.

These future capabilities will be enabled through the CatalogService APIs that have not been currently implemented.

2.4 Results

When a query is executed in ECHO, it automatically generates a Result. Every Result in ECHO has a unique identifier within the system.

A user may execute multiple Queries simultaneously. Result sets from all these queries will be available by name (ResultSetID). These however may be removed from ECHO without notice in a reasonable amount of time, unless they were explicitly saved using the SaveResultSet transaction. Since guests are not allowed to save results in ECHO, the results are not available after the guest completes the ECHO session.

Results can be retrieved by using the Present transaction to issue a PresentRequest XML message as well as through one type of query.

2.5 Catalog Items

A catalog item is any orderable item (granule or collection) in the ECHO system. Retrieving results of a query to ECHO may return several granules or collections, but not all the granules or collections may be orderable. If a granule or collection is orderable, a CatalogItemID (an XML tag in the ECHO DTD) is returned for the granule or collection metadata. Currently, for convenience of client developers, ECHO catalog item ID is in a format like "G123456-GSFC". Every catalog item ID consists of two parts. The first part starts with a capital letter, either a "G" which represents a granule item, or a "C" which represents a collection item. The numerical ID immediately follows the capital letter. The second part is the name of the provider. Two parts are linked with a dash "-". To see the possible required/optional options for a catalog item, you need to run the OrderEntryService "PresentCatalogItem" transaction on that particular catalog item ID. Currently, there is no universal way for a provider to declare which granule or collection is orderable. Each provider must advise ECHO of their order policy or give ECHO the list of granules/collections that are orderable or searchable. Another option is for a provider to follow ECHO's rule to provide orderable notification by giving the price for each orderable item even it is $0.00. Note that if a URL is provided, it is assumed that the client can simply retrieve the data from that URL as a direct link or obtain the data accessing instruction via the URL.

2.6 Orders

An order is a collection of catalog items that a user is interested in, and would presumably order. Each item in the order is associated with a quantity value, and any options available to that item. Within ECHO, a registered user creates an order and then adds, deletes, and updates each item in the order before submitting the order to ECHO. Registered users can look at the status of their orders, as well as the history of all their submitted and shipped orders.

The collection of catalog items that comprises an order does not have to belong to one provider, but can span many providers. When organizing providers and catalog items within an order, another concept called a "provider order" is used. An order can consist of one or more provider orders. Each provider order can consist of one or more catalog items that belong to the same provider. To uniquely identify a particular provider order, one only needs the order ID that the provider order is in, and the provider ID of the provider that is associated with that provider order. When a full order is submitted to ECHO, it is these separate provider orders that are actually submitted to each associated provider.

The OrderEntryService allows registered users to create and change orders, provider orders, or individual catalog items. All transactions within the OrderEntryService deal with orders before they are submitted to the system. Once the "Submit" transaction is executed for a certain order within the OrderEntryService, the user can no longer execute any further changes on that order. However, a user can look at the current and historical status of any of their submitted orders through the order-oriented transactions in the UserAccountService.

Once a provider order is submitted to the appropriate provider, the status of that order can be changed in two ways.

- The provider can send an immediate response, whether they will or will not accept the order, to a provider order submission.
- The provider can wait and asynchronously use the ProviderOrderManagementService API to change the status of an order after they have had time to fully process the order.

2.6.1 Authenticators

While most items that can be ordered through ECHO's data broker are available to the entire user community, some items are restricted. Different providers use different mechanisms to determine if the person ordering restricted data is allowed to access that data. Some providers user a password, others an authentication key (which is a string). To this end, ECHO provides the authenticator mechanism. Providers are allowed to declare what an authenticator looks like to them. Clients can find out this structure, and present it as a template form to the user. The client can then create a named list of authenticators on behalf of the user for each provider using transactions in the user account service. ECHO allows for a registered user to have more than one authenticator stored for a provider since it is possible that different projects that the user represents may have different authenticators. There are transactions for managing this list. Once the list is created, any client can set which one (by name of the authenticator) should be used for a given order. If none is set, then an authenticator is not used. ECHO will send the authenticator along with the order. An order adapter can be used to convert the authenticator from ECHO's format into the information needed for accessing a provider's existing interface if needed.

2.7 Groups

Groups are an aggregating mechanism in ECHO that allows a user to associate a group name with a given set of users. When a group is created, the group's owner specifies an ECHO user to be the group's manager. However, group managers can be added and removed after creation by other group managers. After becoming a member of a group, a user can be granted access to restricted metadata via the data management service.

2.8 Conditions

Conditions represent a partial equation to be evaluated as part of the ACL honoring system. The type of the condition defines the evaluation process. Temporal Conditions use a date range, and the production date of a granule is compared against the date range to check for applicability of the condition. The primary use of conditions is to facilitate reuse among data rules. The same temporal condition can be used by both a restriction and a permission to control access to metadata. To extend a time range, you only have to change one Temporal Condition as opposed to changing multiple data rules. Another type of condition is RestrictionFlag, which can be used to restrict access to granules based on the value of a granule's RestrictionFlag metadata field. For example, this might be used to control access based on a granule's science quality.

2.9 Rules

Rules wrap conditions and provide a complete evaluation capability as part of the ACL honoring system. Rules define which specific data is to be controlled, as well as the condition to use for evaluating if the data should be controlled. Rules also contain a comparator, which is a key part of rule evaluation. Lastly, rules contain data including ActionType (describes which actions the rule applies to), and in the case of a permission, a GroupName (describes which group the permission applies to). Restrictions (one type of a rule) apply to the global ECHO population, and Permissions (the other type of a rule) apply to a specific group.

2.10 Errors

ECHO does not currently have any Error coding system. Therefore, error messages are free text messages. An attempt is made to make the error messages as self explanatory as possible. Sometimes this means that the error messages may contain the name of the user that executed the request resulting in the error message. This makes it impossible to list all the "exact" error messages that are reported. In addition, since there is no finite set of error codes, it is difficult to list every error scenario and the corresponding error message. An attempt is made to cover different types of error messages. If you do encounter a case that is not covered here, please let us know and help us improve on this documentation.

In general, ECHO presents the error message in two major formats. On is the service responding message and another is the response from the Session Manager. Both formats are expressed using XML.

Error Message 1. Example of an ECHO Service responding with an error message.

```
<?xml version="1.0" standalone="no" ?>
<SubscriptionService>
   <CreateSubscriptionResponse>
      <BooleanResult>
         <BooleanResultType>
            <ERROR />
         </BooleanResultType>
         <Message>Guest users aren't allowed access to this transaction.</Message>
      </BooleanResult>
   </CreateSubscriptionResponse>
</SubscriptionService>
```

In the example above, the root tag is the service name that you submitted your request to. The second level tag is the option response with the option that you requested the service to perform. The third level is an expression of Boolean Result; succeed or fail. If the execution of your requested option to the service failed, ERROR is the returned result type. The error message will indicate the detail of the error.

Another example (shown below) is an error responding from the Session Manager. This type of error message is usually returned when the error is detected before the request reached the service layer such as invalid input XML message or the responding service layer cannot handle the returned error message.

Error Message 2. Example of the ECHO Session Manager responding with an error message.

```
<?xml version="1.0" standalone="no" ?>
<SessionManager>
<Error>
   <![CDATA[
      The content of element type "SaveQueryRequest" is incomplete, it must match "(QueryName,QueryExpression)".
   ]]>
</Error>
</SessionManager>
```

The actual error message is listed as ![CDATA[…..]] inside the Error tag.

In addition to the error messages that ECHO sends out, many system level exceptions such as exceptions from Java and from the Oracle database are also sent as the actual error message under certain circumstances.

2.11 Services

2.11.1 Service Interface

A service interface describes a class of extended service that may be offered by an ECHO partner. It is represented in a Web services definition language (WSDL) document and describes the data types and messages that must be supported to implement the service type. Service interface documents (*.wsdl* documents) are maintained by the ECHO system and managed by the ECHO operations team and the ECHO partner community. Information about each service interface is stored in the ECHO UDDI registry and can be accessed through the ECHO UDDI Inquiry interface.

2.11.2 Service Implementation

A service implementation is a physical instantiation of a service interface. The service implementation is represented in a Web services definition language (WSDL) document, which describes the specific Web location and invocation protocol that can be used to invoke the service on the partner's system. The service implementation (*.wsdl* document) imports its data type and message descriptions from a service interface document (that is hosted by ECHO). Information about the service implementation, including the binding information necessary to invoke the service, is maintained in the ECHO UDDI registry and can be accessed through the ECHO UDDI Inquiry Interface.

3 Registration Service

A User registers with the ECHO system by contacting the RegistrationService. The CreateUser transaction in the RegistrationService allows a guest to create a new user account and set up their profile information.

3.1 Transactions

3.1.1 CreateUser

The CreateUser transaction accepts the following fields: UserInformation, AddressInformation (optional), PhoneInformation (optional), and EmailAddress. Contained within UserInformation are fields such as: UserID, UserPassword, FirstName, LastName, and OrganizationName (optional). Currently, ECHO requires a user's password to have at least 10 characters and at least three of the following four types of characters:

1. Upper case letter
2. Lower case letter
3. Digit
4. Any other special characters such as "$", "&", ">", "<", etc.

Once registration is succeeded, UserName and Password can be used to login, and User Account Service APIs can be called.

Code Example 1. Create User.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RegistrationService PUBLIC "-//ECHO RegistrationService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/RegistrationService.dtd">
<RegistrationService>
   <CreateUserRequest>
      <UserName>NewUser1</UserName>
      <Password>1welcomeG$T</Password>
      <UserInformation>
         <FirstName>New</FirstName>
         <LastName>User</LastName>
         <EmailAddress>username@domain</EmailAddress>
         <OptIn>true</OptIn>
      </UserInformation>
   </CreateUserRequest>
</RegistrationService>
```

3.1.2 SubmitProviderApplication

This transaction is used to submit an application to become a provider of data or services for the ECHO clearinghouse. Only a Registered User can issue this request. The applicant should specify an OrganizationName, ProviderContact, a description of holdings (optional) and services (optional) and any additional information they wish to provide. The applicant must provide at least one provider contact, which describes the contact person within the provider's organization in order to process the application. The contact role of provider contact describes the title or position of the contact person. The provider must have a role 'order manager' in order to receive copy of emails to the users about order status update or answer the user's question. Currently, the creation of a provider account is a manual process and is not automated. Once approved to be a provider of ECHO system, the provider can use Provider Account Service to add, delete and update contact information.

Code Example 2. Provider registration application.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RegistrationService PUBLIC "-//ECHO RegistrationService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ RegistrationService.dtd">
<RegistrationService>
  <SubmitProviderApplicationRequest>
    <ProviderApplication>
      <OrganizationName>TEST_ORGANIZATION</OrganizationName>
      <ProviderType><SERVICE/></ProviderType>
      <DiscoveryUrl>
        http://lyta.gsfc.nasa.gov/imsdev-bin/echowhiteboard/private/webwebx.cgi
      </DiscoveryUrl>
      <ProviderContact>
          <ContactRole>admin</ContactRole>
          <ContactFirstName>Lei</ContactFirstName>
          <ContactLastName>Fang</ContactLastName>
          <AddressInformation>
            <AddressID>admin</AddressID>
            <USFormat>TRUE</USFormat>
            <Street1>Shooting Star dr.</Street1>
            <City>GreenBelt</City>
            <State>MD</State>
            <Zip>20770</Zip>
            <Country>USA</Country>
          </AddressInformation>
          <PhoneInformation>
            <PhoneName>phone1</PhoneName>
             <PhoneString>1 301 555 1234 x264</PhoneString>
          </PhoneInformation>
          <EMailAddress>username@domain</EMailAddress>
      </ProviderContact>
      <DescriptionOfHoldings>
           Description of NASA_GSFC data holding is classified, therefore, use default.
      </DescriptionOfHoldings>
      <DescriptionOfServices>
        We Only process the best of the best of the best of the best data....-:)
      </DescriptionOfServices>
      <AdditionalInformation>
        Any additional question, please send to the admin office listed.
      </AdditionalInformation>
    </ProviderApplication>
  </SubmitProviderApplicationRequest>
</RegistrationService>
```

When the application is submitted, ECHO will respond with an XML message that indicates whether the application succeeded and assigns a temporary tracking id that the potential provider can use to communicate with the ECHO Operations team.

However, there is still additional configuration and testing to complete before this ECHO provider is ready to fully participate. Contact the ECHO operations team at echo@killians.gsfc.nasa.gov for details.

4 User Account Service

The User Account Service is used to update information associated with a particular user. This information includes entities such as Addresses, Emails, Phone Numbers, etc. A feature of the ECHO system is that it allows registered users to store multiple addresses, emails, and phone numbers that are distinguished by their unique names. For example, a client may set up an address in his/her profile where the AddressID value is "Work" as well as another address where the AddressID value is "Home". The same is true for phone numbers. The intent here is to allow a registered user to store a number of addresses and be able to choose from them. The API does not explicitly connect these, but a client interface can query for a list of addresses and fill in the blanks in the other API transactions appropriately.

Users (guest or registered user) are not required to store their address information in ECHO. A registered user has the capability of adding any number of addresses to their profile.

4.1 User Account Service Entities

4.1.1 Addresses

Users are not required to submit address information to the ECHO system unless they order data that requires a shipping address, a billing address or a contact address. A registered user has the capability of adding any number of addresses into their profile but is not required to do so.

An ECHO Address entity consists of an address ID, a US format flag, five street address lines, and city, state, zip code, and country fields. The address ID represents a unique name for the address entity like "Home" or "Work". The ECHO Address entity does try to support multiple international mailing formats by using the US format flag. Since international mailing formats can differ greatly from each other, the ECHO Address entity has five Street Address lines that allow for a free format that users and providers can use to fill in any international address in any way that is deemed best. In such a case, the US format flag should be set to “FALSE” and only the first street address and country field are required. However, if the mailing address does follow the normal U.S. mailing standards, then the US format flag should be set to “TRUE” and the city/state/zip-code/country fields will become required fields. In all cases, the country field is required and must comply with the ISO 3-letter country code. For example, “USA” stands for the United States of America while “CAN” stands for Canada.

4.1.2 Emails

Only one email address can be associated with a user account. The rules for an email address in the ECHO system are consistent with global standards for email addresses: username@domain.

4.1.3 Phone Numbers

Phone Number entities are similar to Address entities in that a user may store multiple phone numbers in the ECHO system. Each piece of phone information is given a PhoneName, which is a unique name chosen by the user. An example of a PhoneName is "Home" or "Work". The actual phone number is stored as a string in PhoneString.

4.1.4 Order History

Every order that is placed through the ECHO system is tracked internally. At any time, a user can ask ECHO to present his / her order history. The order history is a listing of all orders the user has transacted during their entire existence in the ECHO system; however, ECHO operations may impose time limits to effectively manage space.

4.1.5 Authenticators

While many items that can be ordered through ECHO’s data broker are available to the entire user community, some items are restricted. Different providers use different mechanisms to determine if the person ordering restricted data is allowed to access that data. Some providers user a password, while others an authentication key (which is a string). To this end, ECHO provides the authenticator mechanism. Providers declare what an authenticator looks like to them. Clients can find out this structure, and present it as a template form to the user. The client can then create a named list of authenticators on behalf of the user for each provider using transactions in the user account service. ECHO allows for a registered user to have more than one authenticator stored for a provider since it is possible that different projects that the user represents may have different authenticators.

4.2 Transactions

4.2.1 ListRoles

The registered user can use this transaction to list the roles associated with oneself. Currently, this list would include all the provider roles and/or admin role associated with this user. The presentation of each role consists of a RoleType, a RoleTarget, and a RoleDescription. RoleType is currently either a PROVIDER or an ADMIN. RoleTarget is used in case of a PROVIDER role to show which provider this user represents. RoleDescription is for the description of the role.

Code Example 3. ListRoles request.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <ListRolesRequest/>
</UserAccountService>
```

Code Example 4. ListRoles response.

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <ListRolesResponse>
      <BooleanResult>
         <BooleanResultType>
            <REQUEST_SUCCEEDED />
         </BooleanResultType>
      </BooleanResult>
      <Role>
         <RoleType>
            <ADMIN />
         </RoleType>
         <roleDescription>Administrator of ECHO</roleDescription>
      </Role>
      <Role>
         <RoleType>
            <PROVIDER />
         </RoleType>
         <roleTarget>LPDAAC_ECS</roleTarget>
         <roleDescription>Provider Access to provider LPDAAC_ECS</roleDescription>
      </Role>
      <Role>
         <RoleType>
            <PROVIDER />
         </RoleType>
         <roleTarget>GSFCECS</roleTarget>
         <roleDescription>Provider Access to provider GSFCECS</roleDescription>
      </Role>
   </ListRolesResponse>
</UserAccountService>
```

4.2.2 *PresentUserInformation*

This transaction allows a user to view his or her information including: first name, last name, email address, and/or organization name.

Code Example 5. PresentUserInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentUserInformationRequest/>
</UserAccountService>
```

4.2.3 *UpdateUserInformation*

This transaction enables a user to update his or her information including: first name, last name, email address, and/or organization name.

Use of this transaction will completely overwrite any previous user information. If an update is submitted that contains blank fields, then existing information in those fields will be overwritten with blanks.

Code Example 6. UpdateUserInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <UpdateUserInformationRequest>
      <UserInformation>
         <FirstName>NewUser</FirstName>
         <LastName>NewOne</LastName>
         <EmailAddress>username@domain</EmailAddress>
         <OptIn>true</OptIn>
      </UserInformation>
   </UpdateUserInformationRequest>
</UserAccountService>
```

4.2.4 *Present Address Information*

A registered user can maintain any number of addresses. Each address will have a name (AddressID) that references the address; all addresses can be added, presented, updated, and deleted by this registered user. There are some address information specifics.

- The name of the address (AddressID) must be unique for the user that the address belongs to and is not case sensitive.
- The US format requires at least one street address line to be filled, as well as, the city, state, zip code and country fields.

- The non-US format only requires one street address line and the country field. The country field must follow the convention of the ISO 3-letter country code.

This transaction allows a registered user to view one or more addresses in this user's profile. For each AddressID specified, the address information of the specified address is presented. Otherwise, if the AdressID is omitted, all of this user's addresses will be presented. However, trying to present a non-existing AddressID will result in an error message.

Code Example 7. PresentAddressInformation transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentAddressInformationRequest>
      <AddressID>work</AddressID>
   </PresentAddressInformationRequest>
</UserAccountService>
```

Code Example 8. PresentAddressInformation transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentAddressInformationRequest/>
</UserAccountService>
```

4.2.5 Add Address Information

This transaction adds one or more addresses to this registered user's profile. Adding an address with an existing AddressID will result in an error message. The AddressID identifies each address. All required address fields (specified in Present Address Information Section) in an address must be filled.

Code Example 9. AddAddressInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <AddAddressInformationRequest>
      <AddressInformation>
         <AddressID>Work</AddressID>
         <USFormat>TRUE</USFormat>
         <Street1>222 bbb st.</Street1>
         <City>future city</City>
         <State>md</State>
         <Zip>22222</Zip>
         <Country>USA</Country>
      </AddressInformation>
   </AddAddressInformationRequest>
</UserAccountService>
```

4.2.6 *UpdateAddressInformation*

This transaction enables a registered user to update one or more addresses in this user's profile. If the user updates multiple addresses and one of them fails, then the whole update fails. Updating an address with an AddressID that had not previously been added will result in an error message. All required address fields in an address must be filled.

Code Example 10. UpdateAddressInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <UpdateAddressInformationRequest>
      <AddressInformation>
         <AddressID>work</AddressID>
         <USFormat>TRUE</USFormat>
         <Street1>9111 Edmonston Rd.</Street1>
         <Street2>Suite 202</Street2>
         <City>Greenbelt</City>
         <State>MD</State>
         <Zip>20770</Zip>
         <Country>USA</Country>
      </AddressInformation>
   </UpdateAddressInformationRequest>
</UserAccountService>
```

4.2.7 *DeleteAddressInformation*

This transaction enables a registered user to delete one or more addresses. If the user deletes multiple addresses and one of them could not be deleted, then the whole deletion fails. Deleting a non-existing address will result in an error message.

Code Example 11. DeleteAddressInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <DeleteAddressInformationRequest>
      <AddressID>work</AddressID>
   </DeleteAddressInformationRequest>
</UserAccountService>
```

4.2.8 *PresentPhoneInformation*

This transaction allows a registered user to present one or more phone numbers in his or her profile. If multiple phone numbers are to be presented and one of them cannot be presented, the whole operation fails. If PhoneName is specified, the phone information of the specified phones is presented; otherwise, all of the phones are presented. However, presenting a non-existing phone number will result in an error message.

Code Example 12. PresentPhoneInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentPhoneInformationRequest>
      <PhoneName>work</PhoneName>
   </PresentPhoneInformationRequest>
</UserAccountService>
```

4.2.9 AddPhoneInformation

This transaction enables a registered user to add one or more phone numbers to a user's profile. All of the phone fields are required and must be filled, or the message is not valid. If multiple phone numbers are added and one of them cannot be added, the transaction fails. However, adding an existing phone number will result in an error message.

Code Example 13. AddPhoneInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <AddPhoneInformationRequest>
      <PhoneInformation>
         <PhoneName>work</PhoneName>
         <PhoneString>1-301-555-8888</PhoneString>
      <PhoneInformation>
   </AddPhoneInformationRequest>
</UserAccountService>
```

4.2.10 UpdatePhoneInformation

This transaction enables a registered user to update one or more phone numbers in their profile. All of the phone fields are required and must be filled, or the message is not valid. If multiple phone numbers are to be updated and one of them cannot be updated, the transaction fails. Updating a non-existing phone number will result in an error message.

Code Example 14. UpdatePhoneInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <UpdatePhoneInformationRequest>
      <PhoneInformation>
         <PhoneName>work</PhoneName>
         <PhoneString>1-301-555-8888</PhoneString>
      </PhoneInformation>
   </UpdatePhoneInformationRequest>
</UserAccountService>
```

4.2.11 DeletePhoneInformation

This transaction allows a registered user to delete one or more phones in his or her profile. If multiple phone numbers are to be deleted and one of them cannot be deleted, the transaction fails. PhoneNames must be specified in order to delete. Deleting non-existing phone numbers will result in an error message.

Code Example 15. DeletePhoneInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <DeletePhoneInformationRequest>
      <PhoneName>work</PhoneName>
   </DeletePhoneInformationRequest>
</UserAccountService>
```

4.2.12 PresentOptionDefinitionsForUser

Users can specify default options (or “preferences”), which ECHO uses during the ordering process. The response in this transaction lists the available options that the user may set:

Table 5: User options.

default_shipping_address	default_contact_address
default_billing_address	default_shipping_phone
default_contact_phone	default_billing_phone
default_shipping_email	default_contact_email
default_billing_email	order_notification_level

Code Example 16. PresentOptionDefinitions transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOptionDefinitionsForUserRequest/>
</UserAccountService>
```

Note: If a particular option is identified in the request message, only information related to that option will appear in the response.

4.2.13 SetOptionSelectionsForUser

Based on the option definitions identified in the response to the PresentOptionDefinitionsForUser transaction, a user can set the options they desire. More information about options can be found in the Order Entry Service section.

Code Example 17. Set option selections for user.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <SetOptionSelectionsForUserRequest>
```

```
<OptionSelection>
   <SimpleOptionSelection>
      <OptionName>default_shipping_address</OptionName>
      <Value>project</Value>
   </SimpleOptionSelection>
</OptionSelection>
<OptionSelection>
   <SimpleOptionSelection>
      <OptionName>default_billing_address</OptionName>
      <Value>work</Value>
   </SimpleOptionSelection>
</OptionSelection>
<OptionSelection>
   <SimpleOptionSelection>
      <OptionName>default_contact_address</OptionName>
      <Value>home</Value>
   </SimpleOptionSelection>
</OptionSelection>
<OptionSelection>
   <SimpleOptionSelection>
      <OptionName>default_shipping_phone</OptionName>
      <Value>project</Value>
   </SimpleOptionSelection>
</OptionSelection>
<OptionSelection>
   <SimpleOptionSelection>
      <OptionName>default_billing_phone</OptionName>
      <Value>work</Value>
   </SimpleOptionSelection>
</OptionSelection>
<OptionSelection>
   <SimpleOptionSelection>
      <OptionName>default_contact_phone</OptionName>
      <Value>home</Value>
   </SimpleOptionSelection>
</OptionSelection>
<OptionSelection>
   <SimpleOptionSelection>
      <OptionName>default_shipping_email</OptionName>
      <Value>project@domain</Value>
   </SimpleOptionSelection>
</OptionSelection>
<OptionSelection>
   <SimpleOptionSelection>
      <OptionName>default_billing_email</OptionName>
      <Value>work@domain</Value>
   </SimpleOptionSelection>
</OptionSelection>
<OptionSelection>
   <SimpleOptionSelection>
      <OptionName>default_contact_email</OptionName>
      <Value>home@domain</Value>
   </SimpleOptionSelection>
```

```
      </OptionSelection>
   </SetOptionSelectionsForUserRequest>
</UserAccountService>
```

4.2.14 PresentOptionSelectionsForUser

This transaction enables a user to view the selections that they have set.

Code Example 18. PresentOptionSelections request.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOptionSelectionsForUserRequest>
      <OptionName>default_contact_address</OptionName>
      <OptionName>default_shipping_address</OptionName>
      <OptionName>default_billing_address</OptionName>
      <OptionName>default_contact_phone</OptionName>
      <OptionName>default_shipping_phone</OptionName>
      <OptionName>default_billing_phone</OptionName>
      <OptionName>default_contact_email</OptionName>
      <OptionName>default_shipping_email</OptionName>
      <OptionName>default_billing_email</OptionName>
   </PresentOptionSelectionsForUserRequest>
</UserAccountService>
```

Code Example 19. Response message for PresentOptionSelectionsForUserResponse.

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
  <PresentOptionSelectionsForUserResponse>
    <BooleanResult>
      <BooleanResultType>
        <REQUEST_SUCCEEDED />
      </BooleanResultType>
    </BooleanResult>
    <OptionSelection>
      <SimpleOptionSelection>
       <OptionName>default_contact_address</OptionName>
       <Value>home</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_shipping_address</OptionName>
        <Value>project</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
```

```
        <OptionName>default_billing_address</OptionName>
        <Value>work</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
       <OptionName>default_contact_phone</OptionName>
       <Value>home</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_shipping_phone</OptionName>
        <Value>project</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_billing_phone</OptionName>
        <Value>work</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
       <OptionName>default_contact_email</OptionName>
       <Value>home@domain</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_shipping_email</OptionName>
        <Value>project@domain</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_billing_email</OptionName>
        <Value>work@domain</Value>
      </SimpleOptionSelection>
    </OptionSelection>
  </PresentOptionSelectionsForUserResponse>
</UserAccountService>
```

4.2.15 Change User Password

This transaction lets the user change his or her password. Currently, ECHO requires users' passwords to have at least 10 but no more than 40 characters; the password must contain at least three of the four types of characters:

1. Upper case letter
2. Lower case letter
3. Digit
4. Any other special characters such as "$", "&", ">", "<", etc.

A regular user must provide the old password for security reasons.

> *Note: An admin user may change any other user's password without providing the old password. However, if the admin user changes his or her own password, the old password is still required.*

Code Example 20. ChangeUserPassword transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <ChangeUserPasswordRequest>
      <OldPassword>1welcomeG$T</OldPassword>
      <NewPassword>newPassword3</NewPassword>
   </ChangeUserPasswordRequest>
</UserAccountService>
```

4.2.15.1 Recall User Name

This transaction lets the user recall his or her username using the e-mail address specified in the username creation process. The username will be sent to the e-mail address which the user specified in the transaction. If there are multiple usernames associated with the e-mail address, all of the usernames will be sent to the user by ECHO mail server.

Code Example 21. Code Example ? RecallUsernameRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService  SYSTEM 'http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd'>
          <UserAccountService>
            <RecallUsernameRequest>
          <EmailString>user1@abcUniversity.edu</EmailString>
 </RecallUsernameRequest>
</UserAccountService>
```

Code Example 22. Code Example? RecallUsernameResponse

```
<?xml version="1.0" standalone="no" ?>
 <!DOCTYPE UserAccountService SYSTEM 'http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd'>
 <UserAccountService>
 <RecallUsernameResponse>
  <BooleanResult>
          <BooleanResultType>
                    <REQUEST_SUCCEEDED />
          </BooleanResultType>
  </BooleanResult>
</RecallUsernameResponse>
</UserAccountService>
```

Email Example :

From: echo@gst.com

To: user1@abcUniversity.edu

Subject: Your ECHO account username

Your username registered in ECHO is: gsfc_user1

4.2.15.2 Reset Password

If a user has forgotten his current password, he may ask ECHO to generate a new password. This new password is then emailed to the user at a specified email address.

Code Example 23. Code Example X Reset User Password

```
<UserAccountService>
        <ResetUserPasswordRequest>
                <UserName>703-User</UserName>
                <EmailString>yourname@xxx.yyy </EmailString>
        </ResetUserPasswordRequest>
</UserAccountService>
```

Email Response Sent to User:

From: Echo System
Sent: Friday, November 04, 2005 3:13 PM
To: yourname@xxx.yyy
Subject: Notification of reset ECHO password

Your password in ECHO has been reset to: pF5La#yWnp%

We recommend you to change this password ASAP, or please contact ECHO operation team at [echo@killians.gsfc.nasa.gov] for further assistance.

4.2.16 PresentOrderHistory

The order APIs in the User Account Service are used to track a user's order history, pending orders and cancelled orders. These APIs apply to orders that have been submitted. Only registered users can use these APIs since they are in User Account Service.

This transaction enables a registered user to view detailed information about user orders that are in terminating order states where orders are no longer pending. The terminating order states are SUBMITTED_WITH_EXCEPTIONS, CANCELLED, CLOSED, or CLOSED_WITH_EXCEPTIONS.

Code Example 24. PresentOrderHistory transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOrderHistoryRequest/>
</UserAccountService>
```

Code Example 25. PresentOrderHistory transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOrderHistoryRequest>
      <TerminatingOrderState>
         <CLOSED/>
      </TerminatingOrderState>
      <TerminatingOrderState>
         <CLOSED_WITH_EXCEPTIONS/>
      </TerminatingOrderState>
   </PresentOrderHistoryRequest>
</UserAccountService>
```

4.2.17 PresentOrderHistorySummary

This transaction is similar to present order history except it just returns the order ID and order state.

Code Example 26. PresentOrderHistorySummary transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOrderHistorySummaryRequest/>
</UserAccountService>
```

Code Example 27. PresentOrderHistorySummary transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOrderHistorySummaryRequest>
      <TerminatingOrderState>
         <CLOSED/>
      </TerminatingOrderState>
      <TerminatingOrderState>
         <CLOSED_WITH_EXCEPTIONS/>
      </TerminatingOrderState>
   </PresentOrderHistorySummaryRequest>
</UserAccountService>
```

4.2.18 PresentPendingOrders

This transaction enables a registered user to view detailed information about user orders that have been submitted but are still pending. The pending order states are SUBMITTING, PROCESSING, PROCESSING_WITH_EXCEPTIONS, or CANCELLING.

Code Example 28. PresentPendingOrder transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentPendingOrdersRequest/>
</UserAccountService>
```

Code Example 29. PresentPendingOrder transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentPendingOrdersRequest>
      <PendingOrderState>
         <SUBMITTING/>
      </PendingOrderState>
      <PendingOrderState>
         <PROCESSING/>
      </PendingOrderState>
   </PresentPendingOrdersRequest>
</UserAccountService>
```

4.2.19 PresentPendingOrderSummary

This transaction is similar to present pending order except it just returns order ID and order state.

Code Example 30. PresentPendingOrderSummary transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentPendingOrderSummaryRequest/>
</UserAccountService>
```

Code Example 31. PresentPendingOrderSummary transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentPendingOrderSummaryRequest>
      <PendingOrderState>
         <SUBMITTING/>
      </PendingOrderState>
      <PendingOrderState>
         <PROCESSING/>
      </PendingOrderState>
   </PresentPendingOrderSummaryRequest>
</UserAccountService>
```

4.2.20 CancelOrder

The registered user may request cancellation of an order or a provider order that has been submitted and has not yet been fulfilled. There may be a cost associated with canceling the order. Either a whole order can be specified (by the first OrderID) or a specific provider order can be specified (by the ProviderOrderID). Specifying both OrderID and ProviderOrderID at the CancelOrderRequest level will result in an error message.

Code Example 32. CancelOrder transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <CancelOrderRequest>
      <ProviderOrderID>
         <ProviderID>1111:0000000</ProviderID>
         <OrderID>0000000</OrderID>
      </ProviderOrderID>
   </CancelOrderRequest>
</UserAccountService>
```

Code Example 33. CancelOrder transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <CancelOrderRequest>
      <OrderID>0000000</OrderID>
   </CancelOrderRequest>
</UserAccountService>
```

4.2.21 Authenticators

The following transactions enable a user to determine the authenticator recognized by a provider, and to create and manage a list of authenticators for use with ordering data from specific providers.

Definitions:

AuthenticatorDefinition:	The structure of an authenticator expected by a provider
AuthenticatorName:	Name a user assigns to a particular authenticator

4.2.21.1 PresentAuthenticatorDefinition

This transaction enables a user to view the authenticator definition specified by the provider.

Code Example 34. PresentAuthenticatorDefinition Request XML message

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService SYSTEM "http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
         <PresentAuthenticatorDefinitionRequest>
```

```
                <ProviderID>GSFCECS</ProviderID>
        </PresentAuthenticatorDefinitionRequest>
</UserAccountService>
```

Code Example 35. Response XML message for PresentAuthenticatorDefinition

```
<?xml version="1.0"?>
<!DOCTYPE UserAccountService SYSTEM "http://api.echo.nasa.gov /echo/dtd/UserAccountService.dtd">
<UserAccountService>
        <PresentAuthenticatorDefinitionResponse>
                <BooleanResult>
                        <BooleanResultType>
                                <REQUEST_SUCCEEDED/>
                        </BooleanResultType>
                </BooleanResult>
                <ProviderID>GSFCECS</ProviderID>
                <Option>
                        <SimpleOption>
                                <Name>authenticator</Name>
                                        <OptionCategory>AUTHENTICATOR_SETTING</OptionCategory>
        <Description>the authenticator string</Description>
                                        <MinOccurs>0</MinOccurs>
                                        <MaxOccurs>1</MaxOccurs>
                                        <Encrypted>True</Encrypted>
                                        <PrimitiveTypeName>
                                                <String/>
                                        </PrimitiveTypeName>
                                </SimpleOption>
                </Option>
        </PresentAuthenticatorDefinitionResponse>
</UserAccountService>
```

4.2.21.2 CreateAuthenticator

This transaction enables a user to create an authenticator that matches a provider authenticator definition.

Code Example 36. CreateAuthenticator Request message:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService SYSTEM "http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
        <CreateAuthenticatorRequest>
                <ProviderID>GSFCECS</ProviderID>
                <AuthenticatorName>MyGSFC_auth</AuthenticatorName>
                <OptionSelection>
                        <SimpleOptionSelection>
                                <OptionName>authenticator</OptionName>
                                        <Value>b6a5d0c8b2262e6392d49ce9567c1677</Value>
```

```
                    </SimpleOptionSelection>
            </OptionSelection>
    </CreateAuthenticatorRequest>
</UserAccountService>
```

4.2.21.3 PresentAuthenticator

This transaction enables a user to present an authenticator created by the user.

Code Example 37. PresentAuthenticator Request XML message

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService SYSTEM "http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
        <PresentAuthenticatorRequest>
                <ProviderID>GSFCECS</ProviderID>
                <AuthenticatorName>MyGSFC_auth</AuthenticatorName>
        </PresentAuthenticatorRequest>
</UserAccountService>
```

Code Example 38. Response XML message for PresentAuthenticator Request:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE UserAccountService SYSTEM "http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
        <PresentAuthenticatorResponse>
                <BooleanResult>
                        <BooleanResultType>
                                <REQUEST_SUCCEEDED/>
                        </BooleanResultType>
                </BooleanResult>
                <ProviderID>GSFCECS</ProviderID>
                <AuthenticatorName>MyGSFC_auth</AuthenticatorName>
                <OptionSelection>
                        <SimpleOptionSelection>
                                <OptionName>authenticator</OptionName>
                                        <Value>b6a5d0c8b2262e6392d49ce9567c1677</Value>
                        </SimpleOptionSelection>
                </OptionSelection>
        </PresentAuthenticatorResponse>
</UserAccountService>
```

4.2.21.4 DeleteAuthenticator

This transaction enables a user to delete authenticators created by the user.

Code Example 39. DeleteAuthenticator Request XML message

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService SYSTEM "http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
```

```
        <DeleteAuthenticatorRequest>
                <ProviderID>GSFCECS</ProviderID>
                <AuthenticatorName>MyGSFC_auth</AuthenticatorName>
        </DeleteAuthenticatorRequest>
</UserAccountService>
```

4.2.21.5 UpdateAuthenticator

This transaction enables a user to update an authenticator created by the user.

Code Example 40. UpdateAuthenticator Request XML message

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService SYSTEM "http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
        <UpdateAuthenticatorRequest>
                <ProviderID>GSFCECS</ProviderID>
                <AuthenticatorName>MyGSFC_auth</AuthenticatorName>
                <OptionSelection>
                        <SimpleOptionSelection>
                                <OptionName>authenticator</OptionName>
        <Value>b6a5d0c8b2262e6392d49ce9567c1688</Value>
                        </SimpleOptionSelection>
                </OptionSelection>
        </UpdateAuthenticatorRequest>
</UserAccountService>
```

4.2.21.6 ListAuthenticatorNames

This transaction enables a user to list authenticators created by the user.

Code Example 41. ListAuthenticatorNames Request XML message

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService SYSTEM "http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
        <ListAuthenticatorNamesRequest>
                <ProviderID>GSFCECS</ProviderID>
        </ListAuthenticatorNamesRequest>
</UserAccountService>
```

Code Example 42. Response XML message for ListAuthenticatorNames Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService SYSTEM “http://api.echo.nasa.gov/echo/dtd/UserAccountService.dtd”>
<UserAccountService>
        <ListAuthenticatorNamesResponse>
                <BooleanResult>
                <BooleanResultType>
                <REQUEST_SUCCEEDED/>
                </BooleanResultType>
                </BooleanResult>
                <ProviderID>GSFC-TEST</ProviderID>
                <AuthenticatorName>MyGSFC_auth</AuthenticatorName>
        </ListAuthenticatorNamesResponse>
</UserAccountService>
```

4.3 Error Messages

The following table gives the error messages associated with each transaction in the User Account Service.

Table 6: User Account Service error messages.

Transaction	Error Message	Description
Add Address Information	This address already exists. Please add a new one	This error is returned when the address already exists.
Update Address Information	This address does not exist.	This error is returned if the address indicated for update does not exist.
Present Address Information	<addressID> is not found.	This error is returned when trying to request for an address that does not exist.
Delete Address Information	<addressID> is not found.	This error is returned when trying to delete an address that does not exist.
Add Phone Information	This phone number already exists. Please add a new one	This error is returned when trying to add a phone using a name that already exists.
Update Phone Information	This phone number does not exist.	This error is returned when trying to update a phone that does not exist.
Present Phone Information	EJB Exception: : java.lang.NullPointerException at	This error is returned when requesting for a phone using phone name that does not exist.
Delete Phone Information	<phoneName> is not found.	This error is returned when trying to delete a phone that does not exist.
Change User Password	The old password is incorrect.	This error is returned when entering an incorrect current password for password changing.
Recall User Name	Invalid e-mail address	A guest user uses an invalid e-mail address to retrieve a username
	no username is found associated with specified email address user1@abcUniversity.edu	A guest user uses a valid e-mail address to retrieve a non-existing username

Transaction	Error Message	Description
	Submission of the E-mail to address "xxx@yyy" failed due to ECHO mail server errors, Please retry. If this condition persists, please contact the ECHO operations at echo@killians.gsfc.nasa.gov."	Email transmission fails at transmission time. i.e. ECHO mail server has a problem.
Cancel Order	Unable to locate order <orderID>.	This error is returned when trying to cancel an order that does not exist for this user.
	The provider <ProviderID> does not exist in the order <OrderID>.	This error is returned when the provider ID indicated is not included in the order referred by the order ID.
	Cancellation of the order from provider:<ProviderID> has already been rejected by the provider.	This error is returned when trying to again cancel an order that cancellation on this order was rejected by the provider once before.
	Provider Order from <ProviderID> is in an invalid state to be cancelled.	This error is returned when user trying to cancel an order that is placed in a state that cannot be cancelled.
Set Option Selections	[<OptionName>] is not a valid property for the item.	This error is returned when trying to access an option that is not valid for the item.
	There is no property <ChildOption> defined for the type: <ParentOption>	This error is returned when submitting an invalid option structure.
Create authenticator	"This authenticator [" + authenticatorName + "] already exists, please choose another name"	This error is returned if the authenticator name is already exist.
	"The authenticator value that user is trying to set does not match provider's authenticator definition"	This error is returned when the authenticator user created is not valid against the provider's authenticator definition.
Delete authenticator	"This authenticator [" + authenticatorName[i] + "] does not exist, please choose another name"	This error is returned when user deletes an authenticator with a wrong authenticator name.
Update authenticator	"This authenticator [" + authenticatorName + "] does not exist, please choose another name"	This error is returned when user updates an authenticator with a wrong authenticator name.
Present authenticator	"This authenticator [" + authenticatorName + "] does not exist, please choose another name"	This error is returned when user presents a authenticator with an wrong name.

5 Group Management Service

The Group Management Service is used to aggregate users together to form groups. Each group created within ECHO has a specific owner. The owner has the option of managing the group or delegating management of the group to one or more other registered users in the system. Any user can manage a group; the user does not have to have the provider role. However, only users who have the provider role can create a group.

Group managers are the curators of the group and are responsible for maintaining the group's description, name, membership and management roster. A manager has absolute control over all of the previous attributes of the group. As such, a group's management team should be chosen carefully to prevent misuse.

The following conditions also apply:

1. The individual user should be a registered user.
2. The owner of the group needs to add himself as a manager before he can manage the group. Note: Adding a user as the group manager does not make them a member of the group. So, if the group manager should also have the privileges of the group, the AddMember transaction should be called for that user.

Groups also provide a communication mechanism. Any user in the ECHO system has the capability to contact a group's management team. Similarly, any manager of a group has the capability to contact the group's members. This could be useful for testers to inform their test team when to suspend testing.

The GroupManagementService provides a mechanism for aggregating users together. The groups created from this service are used in the Data Management Service.

5.1 Transactions

5.1.1 ListGroups

The ListGroups transaction will display all the groups managed by the invoking user.

Code Example 43. ListGroups transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <ListGroupsRequest/>
</GroupManagementService>
```

5.1.2 ListAllGroups

The ListAllGroups transaction returns a list of all the group names that exist in the system.

Code Example 44. ListAllGroups transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <ListAllGroupsRequest></ListAllGroupsRequest>
</GroupManagementService>
```

5.1.3 PresentGroupInformation

The PresentGroupInformation transaction displays a group's name, description, and its list of managers.

Code Example 45. PresentGroupInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <PresentGroupInformationRequest>
      <GroupName>Group1</GroupName>
   </PresentGroupInformationRequest>
</GroupManagementService>
```

5.1.4 CreateGroup

The example below creates a group named 'MODIS QA' and identifies Tester1 as the manager of the group. It is important to note that group names must be globally unique. Providers that create groups should therefore be judicious in the names they select.

Code Example 46. CreateGroup transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <CreateGroupRequest>
      <GroupInformation>
         <GroupName>MODIS QA</GroupName>
         <description>
            Test team assembled to test MODIS metadata.
         </description>
         <GroupManager>
            <UserName>Tester1</UserName>
         </GroupManager>
      </GroupInformation>
   </CreateGroupRequest>
</GroupManagementService>
```

5.1.5 UpdateGroupInformation

The UpdateGroupInformation transaction can be used to update a group's description as well as its entire management team. Although we provide AddManager and RemoveManager transactions for granular control of the group's management team, the UpdateGroupInformation transaction is used to completely re-write the management team. This might be useful if a group managed by contractors from one company is handed over to another company for management.

Code Example 47. UpdateGroupInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <UpdateGroupInformationRequest>
      <GroupInformation>
            <GroupName>MODIS QA</GroupName>
            <description>
            Test team assembled to test MODIS metadata
            </description>
         <GroupManager>
            <UserName>Tester2</UserName>
         </GroupManager>
      </GroupInformation>
   </UpdateGroupInformationRequest>
</GroupManagementService>
```

In the above example, the description of the group has changed to indicate a change in the team handling MODIS metadata testing. Similarly, Tester2 is the exclusive manager of the group. After this transaction is invoked, Tester1 is no longer a manager of the 'MODIS QA' group. If Tester1 attempts to invoke a transaction against 'MODIS QA' that requires management privileges they will be rejected.

5.1.6 RenameGroup

The RenameGroup transaction will rename a group. The system will send an e-mail to all group members about the change if Notify is true.

Code Example 48. RenameGroup transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <RenameGroupRequest>
      <OldGroupName>Group1</OldGroupName>
      <NewGroupName>Group2</NewGroupName>
      <Notify>true</Notify>
   </RenameGroupRequest>
</GroupManagementService>
```

5.1.7 DestroyGroup

This transaction disbands a group. The system will send an e-mail to all group managers and members if Notify is true.

Code Example 49. DestroyGroup transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
  <DestroyGroupRequest>
    <GroupName>Group1</GroupName>
    <Notify>true</Notify>
  </DestroyGroupRequest>
</GroupManagementService>
```

5.1.8 ListMembers

The ListMembers transaction will list all the members of a group. Managers are not included in the list unless they have been added as group members also.

Code Example 50. ListMembers transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
  <ListMembersRequest>
  <GroupName>Group1</GroupName>
  </ListMembersRequest>
</GroupManagementService>
```

5.1.9 AddMember

The AddMember transaction will add a member to the group. The system will send an e-mail to the newly-added member if Notify is true.

Code Example 51. AddMember transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
  <AddMemberRequest>
    <GroupName>Group1</GroupName>
    <UserName>billg</UserName>
    <Notify>true</Notify>
  </AddMemberRequest>
</GroupManagementService>
```

5.1.10 ContactMembers

In addition to aggregating users, groups provide a communication mechanism. Any user in the ECHO system has the capability to contact a group's management team. Similarly, any manager of a group has the capability to contact the group's members. This could be useful for testers to inform their test team when to suspend testing.

Code Example 52. ContactMembers transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <ContactMembersRequest>
      <GroupName>MODIS QA</GroupName>
      <GroupMessage>
         <MessageFormat> <TXT/> </MessageFormat>
         <payload>
         Test Team-
         Stop testing on Friday at 5pm and resume testing Monday morning
         at 8am.  We will have a meeting at noon on Monday to discuss progress.
         Have a great weekend!
         - Your Boss
         </payload>
      </GroupMessage>
   </ContactMembersRequest>
</GroupManagementService>
```

The example above covers the ContactMembers transaction. This transaction can only be invoked by a group manager and is used to relay important information to the group's membership roster. The only supported MessageFormat is TXT.

5.1.11 IsMember

The IsMember transaction checks if a user is a member of the specified group.

Code Example 53. IsMember transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <IsMemberRequest>
      <GroupName>Group1</GroupName>
      <UserName>billg</UserName>
   </IsMemberRequest>
</GroupManagementService>
```

5.1.12 RemoveMember

This transaction removes a member from a group. The system will send an e-mail to the removed member if Notify is true.

Code Example 54. RemoveMember transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <RemoveMemberRequest>
      <GroupName>Group1</GroupName>
      <UserName>linust</UserName>
      <Notify>true</Notify>
   </RemoveMemberRequest>
</GroupManagementService>
```

5.1.13 RemoveAllMembers

This transaction removes all members from a group. The list of managers is not affected, so if a member was both a member and a manager, that member will still be a member after this transaction is completed. The system will send an e-mail to all removed members if Notify is true.

Code Example 55. RemoveAllMembers transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <RemoveAllMembersRequest>
      <GroupName>Group1</GroupName>
      <Notify>true</Notify>
   </RemoveAllMembersRequest>
</GroupManagementService>
```

5.1.14 ListManagers

The ListManagers transaction lists all the managers of a group.

Code Example 56. ListManagers transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
  <ListManagersRequest>
    <GroupName>Group1</GroupName>
  </ListManagersRequest>
</GroupManagementService>
```

5.1.15 AddManager

This transaction adds a manager to the group. The new manager is not required to be a member of the group. To add a manager to the list of members of a group, use the AddMember transaction (see Section 5.1.9). The system will send an e-mail to the new manager if Notify is true.

Code Example 57. AddManager transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <AddManagerRequest>
      <GroupName>Group1</GroupName>
      <UserName>billg</UserName>
      <Notify>true</Notify>
   </AddManagerRequest>
</GroupManagementService>
```

5.1.16 ContactManagers

This transaction is used to send a message to a group's managers. Unlike the ContactMembers transaction, any user in the ECHO system may invoke the ContactManagers transaction. This could be used by a user who wishes to become a member of a group and would like to open a dialog with the management team of that group.

Code Example 58. ContactManagers transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
  <ContactManagersRequest>
    <GroupName>Group1</GroupName>
    <GroupMessage>
      <MessageFormat>
        <TXT/>
      </MessageFormat>
      <payload>I want to join your group.</payload>
    </GroupMessage>
  </ContactManagersRequest>
</GroupManagementService>
```

5.1.17 RemoveManager

The RemoveManager transaction removes a manager from a group. A manager cannot be removed if that person is the group's only manager.

Code Example 59. RemoveManager transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v6.0)// EN"
"http://api.echo.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <RemoveManagerRequest>
      <GroupName>Group</GroupName>
      <UserName>stinky2</UserName>
      <Notify>true</Notify>
   </RemoveManagerRequest>
</GroupManagementService>
```

5.2 Error Messages

All errors caused by invoking transactions within the Group Management Service result in a rollback condition. That is, if you pass N groups to the CreateGroup transaction, and ECHO fails to create the Nth group, no groups are created. This behavior is consistent with the rest of the ECHO system. The GroupManagementService is available for use by providers and users. Guests are not permitted to invoke any transaction in the service.

The following table gives the possible error messages associated with each transaction within the Group Management Service.

Table 7: GMS error messages.

Transaction	Error Message	Description
AddManager AddMember ContactManagers ContactMembers DestroyGroup IsMember ListManagers ListMembers	Group specified does not exist.	The user specified a group that does not exist.

Transaction	Error Message	Description
PresentGroupInformation RemoveAllMembers RemoveManager RemoveMember RenameGroup UpdateGroupInformation		
AddManager AddMember RemoveManager RemoveMember RenameGroup UpdateGroupInformation	User' <username>' is not a manager of a group. Only group managers can add other group managers.	The user that invoked the transaction is not a manager of the group.
AddManager AddMember RemoveManager RemoveMember	User specified does not exist.	The user specified a user that does not exist.
AddManager AddMember	User specified is already a manager.	The user specified a user that is already a manager.
ContactManagers ContactMembers	No message was present.	The user specified a blank message.
CreateGroup	Only providers are able to create groups.	The user that invoked the transaction was a regular user (and a provider is the only person permitted to invoke this transaction).
	Manager specified does not exist.	The user specified a Manager that does not exist.
	<The missing item> was not specified.	The user specified a blank name, Cause, or manager username.
	A group with the name <the name> already exists.	The user specified a group name that is already in use.
RenameGroup	The new group name is already in use.	The 'NewGroup' specified already exists.
	Group specified does not exist.	The 'OldGroup' specified does not exist.
UpdateGroupInformation	Group specified does not have a description.	The user specified a blank description.
	The user specified does not exist.	The user specified manager(s) that do not exist.
DestroyGroup	Only managers of a group are permitted to destroy the group (and you are not a manager of this group).	The user invoking the transaction is not a manager of the group.
IsMember	User does not exist in the system.	The user specified a user that does not exist.
ListMembers	Only managers of a group are permitted to list the group's members (and you are not a manager of this group).	The user invoking the transaction is not a manager of the group.

Transaction	Error Message	Description
RemoveAllMembers	You are not a manager of the group.	The user invoking the transaction is not a manager of the group.
RemoveManager	User specified is not a manager.	The user specified a user that is not a manager.

6 Client Interface

This section describes the Client Partner API, which allows for various client types including user-interactive, metadata harvesting and other batch processing. Most ECHO clients provide access to ECHO's Earth Science metadata and browse catalog and order broker, while other client objectives exist such as Data Partner facilitation and metrics collection. Client Partner APIs can be found on the ECHO Web site and are described in the following sections.

6.1 Catalog Service

This service allows users to query ECHO's metadata clearinghouse that contains catalog items such as data sets ("collections") and granules. For example, ECHO allows users to search on

- Campaign or project (e.g., TRMM, IAA Environmental Program)
- Data set
- Date
- Sensor (e.g., Acoustic Sounders, CO2 Analyzers)
- Source or platform (e.g., Aircraft, TERRA, Apollo, Gravity Stations)
- Geographic area
- Time periods

The query is written in the IIMS Alternative Query Language (AQL).

This service also supports the storage and execution of named queries and the storage of named results.

6.1.1 Transactions

6.1.1.1 ExplainCollection

`This transaction is not yet implemented.`

6.1.1.2 ListSavedQueries

This message is used to request a list of the names of the saved queries. Guests do not have access to this transaction.

6.1.1.3 ListSavedResultSets

This message is used to request a list of the names of the saved result sets. Guests do not have access to this transaction.

Code Example 60. List Saved ResultSets transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<!-- Description:  Lists saved result sets. -->
<CatalogService>
   <ListSavedResultSetsRequest/>
</CatalogService>
```

6.1.1.4 Present

To retrieve the results of a previously executed query to ECHO, the Present transaction of the CatalogService should be used to send a PresentRequest XML message to ECHO. This message mainly consists of the ResultSetID that is returned as part of the response to the previously executed query and the results presentation specification.

The following is a sample PresentRequest message to return results from a search for granules. Note that the ResultSetID value should match the value you get from the QueryResponse message.

Code Example 61. Present API transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <PresentRequest>
      <ResultSetID>RU315021139942533963</ResultSetID>
      <PresentationDescription>
         <TupleType>
            <attributeName>GranuleUR</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>GranuleVersionId</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>sizeMBDataGranule</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>price</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>ECHOItemId</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>

         <TupleType>
           <attributeName>RangeBeginningDate</attributeName>
           <PrimitiveTypeName>
              <String/>
           </PrimitiveTypeName>
         </TupleType>

         <TupleType>
            <attributeName>ShortName</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>

         <TupleType>
            <attributeName>DayNightFlag</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>

         <TupleType>
            <attributeName>ECHOInsertDate</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>

         <TupleType>
            <attributeName>Sensor</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>

         <DTDType>
            <ECHO/>
         </DTDType>
      </PresentationDescription>
      <IteratorSize>5</IteratorSize>
      <Cursor>1</Cursor>
   </PresentRequest>
</CatalogService>
```

ResultSetID specifies the result set that should be presented. For registered users the last executed query-result could be from a previous session. For guests, the ResultSetID must be from the same session. This field is a mandatory field. Nominally, a result set is only available until the next query is performed. A client program can explicitly save the result set for later reference if it will be needed.

6.1.1.4.1 Query Results Presentation Specification

The result presentation specification can be used to specify the subset of the information that should be returned as a result of a query. The description below is applicable to both PresentRequest message and QueryRequest when requesting results as part of QueryResponse message.

The following elements are used to specify the format and content of the results of a query:

- **MessageFormat.** Specifies the format of the results. It takes values XML, HTML, TXT. Currently ECHO only supports XML.
- **Cursor.** Specifies the first record to be returned. (e.g., a value of 5 will return results starting from the 5th record. If none is specified it defaults to 1).
- **IteratorSize.** Specifies the number of results to be returned. (e.g., a value of 10 combined with the Cursor value of 5 will return results 5 through 14. For one transaction, the maximum number of results able to be returned is 2000).
- **PresentationDescription.** Defines the format of the results. It is divided into 3 parts. DTDType and TupleTypes and PredefinedPresentationType. The DTDType defines the structure of the results. In case of

XML MessageFormat, DTDType is used to specify that the XML results are to be returned in accordance to what DTD. The DTDType defaults to <ECHO/>.

The other elements SortField, QueryScope, CollectionName, and CatalogEntryType have been added for possible future requirements, but they are not currently implemented.

6.1.1.4.2 PresentationDescription

The two parts in the PresentationDescription together determine the XML that is returned. As mentioned above the DTDType determines the DTD the results will conform to.

The DTDs for granules and collections are separate to eliminate any inconsistencies due to overlapping elements with different definitions.

The DTD for granules is located at http://api.echo.nasa.gov/echo/dtd/ECHOGranuleResults.dtd

The DTD for collections is located at http://api.echo.nasa.gov/echo/dtd/ECHOCollectionResults.dtd

If the TupleType elements are not specified, all the elements that can be returned for the specified DTDType will be returned. This may be time and resource consuming. If not all the elements are required, then a subset of these elements can be specified in the TupleTypes and only those will be returned.

TupleType takes the name of the result attribute (element) in the attributeName element. It also has an element PrimitiveTypeName that specifies the data type of the attribute. Currently, PrimitiveTypeName is ignored.

The list of possible result attributes (elements) that can be specified in TupleTypes depends on the DTD to which the results will conform. For example, if the results conform to the DTD for granules listed above then the elements that can be specified are only all XML elements defined in this DTD under and including GranuleURMetaData.

Whatever element is specified in the TupleType, all the elements under it in the hierarchy will also be returned in the results. For example, if Platform is specified, Platform, PlatformShortName, Instrument, InstrumentShortName, Sensor, SensorShortName, SensorCharacteristitcs, SensorCharacteristicName, SensorCharacteristicValue and OperationMode, will also be returned.

If only Sensor is specified, then all the key elements above it in the hierarchy – InstrumentShortName, Instrument, PlatformShortName and Platform, GranuleUR and GranuleURMetaData – will also be returned. This is to ensure that the data is identified correctly.

Please note for spatial information BoundingBox, Polygon etc., CANNOT be used in the TupleType. Only the element enclosing the spatial bounding rectangle/polygon can be specified in the TupleType. For example, for ECHO format results for Granules, HorizontalSpatialDomainContainer should be specified to get spatial information.

The spatial elements identified in the following table cannot be specified as TupleTypes:

Table 8: Spatial elements that cannot be specified as TupleTypes.

Point	Circle	BoundingRectangle
GPolygon	Polygon	PointLongitude
PointLatitude	CenterLatitude	CenterLongitude
Radius	WestBoundingCoordinate	NorthBoundingCoordinate
EastBoundingCoordinate	SouthBoundingCoordinate	Boundary
ExclusiveZone	SinglePolygon	MutiPolygon
OutRing	InnerRing	

Specifying GranuleURMetaData as a TupleType to be returned is equivalent to not specifying any TupleTypes, as the result is the same as returning all the elements in the result DTD.

PredefinedPresentationType is not currently implemented. It will be used in the future to specify an ECHO defined name that represents a predetermined set of TupleTypes. Currently this field is ignored.

If PresentationDescription is not specified at all, it defaults to DTDType to be <ECHO/> and no TupleTypes.

An example QueryRequest with nothing specified for PresentationDescription (taking default values) and its response with results follows. Note that the actual result string is embedded in a wrapper XML message within a CDATA field, so that even though it looks like XML, it will have to be extracted in order to parse it as XML. This is done to accommodate non-XML return formats.

Code Example 62. QueryRequest with default PresentationDescription.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
     <QueryRequest>
          <QueryExpression>
               <query>
<![CDATA[

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
     <for value="granules"/>
          <dataCenterId>
               <value>ORNL_TS1</value>
          </dataCenterId>
     <where>
          <granuleCondition>
               <dataSetId><textPattern>'%SAT%'</textPattern></dataSetId>
          </granuleCondition>
     </where>
</query>

]]>

               </query>
               <namespace>none</namespace>
               <QueryLanguage>
                    <IIMSAQL/>
               </QueryLanguage>
          </QueryExpression>
          <ResultType>
               <RESULTS/>
          </ResultType>
          <IteratorSize>2</IteratorSize>
     </QueryRequest>
</CatalogService>
```

Code Example 63. Query response.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
```

```
<CatalogService>
     <QueryResponse>
          <BooleanResult>
               <BooleanResultType>
                    <REQUEST_SUCCEEDED/>
               </BooleanResultType>
          </BooleanResult>
          <ReturnData>
          <MessageFormat>
               <XML/>
          </MessageFormat>
          <payload>
               <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "-//ECHO results (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/ECHOGranuleResults.dtd">
<results>
     <provider name='ORNL_TS1'>
          <result itemId='G712283-ORNL' number='1'>
               <GranuleURMetaData>
                    <ECHOItemId>G71283-ORNL_TS1</ECHOItemId>
                    <GranuleUR>BOREAS_RADARSAT.data_readme</GranuleUR>
                    <ECHOInsertDate>2001-05-15 16:59:37.0</ECHOInsertDate>
                    <ECHOLastUpdate>2001-05-15 16:59:46.0</ECHOLastUpdate>
                    <GranuleShortName>data_readme</GranuleShortName>
                    <AccessConstraint>PUBLIC</AccessConstraint>
                    <CollectionMetaData>
                         <ShortName>BOREAS RADARSAT IMAGES CD-ROM</ShortName>
                         <VersionID>0</VersionID>
                         <DataSetId>BOREAS RADARSAT IMAGES CD-ROM</DataSetId>
                    </CollectionMetaData>
                    <DataGranule>
                         <SizeMBDataGranule>66</SizeMBDataGranule>
                    </DataGranule>
                    <SpatialDomainContainer>
                         <HorizontalSpatialDomainContainer>
                              <BoundingRectangle>
                                   <WestBoundingCoordinate>-111</WestBoundingCoordinate>
                                   <NorthBoundingCoordinate>60</NorthBoundingCoordinate>
                                   <EastBoundingCoordinate>-93</EastBoundingCoordinate>
                                   <SouthBoundingCoordinate>49</SouthBoundingCoordinate>
                              </BoundingRectangle>
                         </HorizontalSpatialDomainContainer>
                    </SpatialDomainContainer>
                    <MeasuredParameter>
                         <MeasuredParameterContainer>
                              <ParameterName>RADAR BACKSCATTER</ParameterName>
                         </MeasuredParameterContainer>
                    </MeasuredParameter>
                    <StorageMediumClass>
                         <StorageMedium>On-Line</StorageMedium>
                    </StorageMediumClass>
                    <Platform>
                         <PlatformShortName>RADARSAT-1</PlatformShortName>
```

```
                <Instrument>
                    <InstrumentShortName>SAR</InstrumentShortName>
                    <Sensor>
                        <SensorShortName>SAR</SensorShortName>
                    </Sensor>
                </Instrument>
            </Platform>
            <Campaign>
                <CampaignShortName>BOREAS</CampaignShortName>
            </Campaign>
            <Contact>
                <Role>User Services Office</Role>
                <ContactOrganizationName>ORNL DAAC User Services Office</ContactOrganizationName>
                <ContactOrganizationAddress>
                    <StreetAddress>Oak Ridge National Laboratory,      P.O. Box 2008, MS 6407</StreetAddress>
                    <City>Oak Ridge</City>
                    <StateProvince>Tennessee</StateProvince>
                    <PostalCode>37831-6407</PostalCode>
                    <Country>USA</Country>
                </ContactOrganizationAddress>
                <OrganizationTelephone>
                    <TelephoneNumber>+1 (865) 241-3952</TelephoneNumber>
                    <TelephoneType>Telephone</TelephoneType>
                </OrganizationTelephone>
                <OrganizationEmail>
                    <ElectronicMailAddress>ornldaac@ornl.gov</ElectronicMailAddress>
                </OrganizationEmail>
            </Contact>
            <OnlineAccessURL>http://daac.ornl.gov/boreas/GRAB_BAG/radarsat/data/data_readme</OnlineAccessURL>
        </GranuleURMetaData>
    </result>

    <result number='2' itemId='G108456-ORNL_TS1'>
        <GranuleURMetaData>
            <ECHOItemId> G108456-ORNL_TS1</ECHOItemId>
            <GranuleUR>FIFE_SAT_AVHR.7034fife.avh</GranuleUR>
            <ECHOInsertDate>2001-05-15 22:51:57.0</ECHOInsertDate>
            <ECHOLastUpdate>2001-05-15 22:52:05.0</ECHOLastUpdate>
            <GranuleShortName>7034fife.avh</GranuleShortName>
            <AccessConstraint>PUBLIC</AccessConstraint>
            <CollectionMetaData>
                <ShortName>SATELLITE AVHRR EXTRACTED DATA (FIFE)</ShortName>
                <VersionID>0</VersionID>
                <DataSetId>SATELLITE AVHRR EXTRACTED DATA (FIFE)</DataSetId>
            </CollectionMetaData>
            <DataGranule>
                <SizeMBDataGranule>10000</SizeMBDataGranule>
            </DataGranule>
            <SpatialDomainContainer>
                <HorizontalSpatialDomainContainer>
                    <BoundingRectangle>
                        <WestBoundingCoordinate>-96.625</WestBoundingCoordinate>
                        <NorthBoundingCoordinate>39.125</NorthBoundingCoordinate>
```

```
                                        <EastBoundingCoordinate>-96.625</EastBoundingCoordinate>
                                        <SouthBoundingCoordinate>39.125</SouthBoundingCoordinate>
                                    </BoundingRectangle>
                                </HorizontalSpatialDomainContainer>
                            </SpatialDomainContainer>
                            <MeasuredParameter>
                                <MeasuredParameterContainer>
                                    <ParameterName>IRRADIANCE REFLECTANCE</ParameterName>
                                </MeasuredParameterContainer>
                            </MeasuredParameter>
                            <StorageMediumClass>
                                <StorageMedium>On-Line</StorageMedium>
                            </StorageMediumClass>
                            <Platform>
                                <PlatformShortName>NOAA-9 NOAA-10 NOAA-11</PlatformShortName>
                                <Instrument>
                                    <InstrumentShortName>AVHRR</InstrumentShortName>
                                    <Sensor>
                                        <SensorShortName>AVHRR</SensorShortName>
                                    </Sensor>
                                </Instrument>
                            </Platform>
                            <Campaign>
                                <CampaignShortName>FIFE</CampaignShortName>
                            </Campaign>
                                <Contact>
                                    <Role>User Services Office</Role>
                                    <ContactOrganizationName>ORNL DAAC User Services Office</ContactOrganizationName>
                                    <ContactOrganizationAddress>
                                        <StreetAddress>Oak Ridge National Laboratory,      P.O. Box 2008, MS
6407</StreetAddress>
                                        <City>Oak Ridge</City>
                                        <StateProvince>Tennessee</StateProvince>
                                        <PostalCode>37831-6407</PostalCode>
                                        <Country>USA</Country>
                                    </ContactOrganizationAddress>
                                    <OrganizationTelephone>
                                        <TelephoneNumber>+1 (865) 241-3952</TelephoneNumber>
                                        <TelephoneType>Telephone</TelephoneType>
                                    </OrganizationTelephone>
                                    <OrganizationEmail>
                                        <ElectronicMailAddress>ornldaac@ornl.gov</ElectronicMailAddress>
                                    </OrganizationEmail>
                                </Contact>
                            <OnlineAccessURL>http://daac.ornl.gov/fife_1/data/sat_obs/sat_avhr/y1987/7034fife.avh</OnlineAccessURL>
                        </GranuleURMetaData>
                    </result>
                </provider>
</results>]]>
        </payload>
        </ReturnData>
        <RequestID>RGuest7213396921014333628487</RequestID>
        <ResultSetID>RGuest7213396921014333628487</ResultSetID>
        <ResultType>
```

```
                <RESULTS/>
            </ResultType>
            <Status>
                <SUCCESS_RESULTS_AVAILABLE/>
            </Status>
           <Hits>85</Hits>
           <Cursor>3</Cursor>
     </QueryResponse>
</CatalogService>
```

The DTD for the result (as noted in the XML itself) is located at:

http://api.echo.nasa.gov/echo/dtd/ECHOGranuleResults.dtd

The root element, <results> begins every result XML document. The results are grouped per provider. Hence the element under result is provider with an attribute name that specifies the name of the provider.

6.1.1.5 GetMetadata

Users have flexibility of getting metadata information without providing a resultSetID that was obtained from the previous Query API transaction. Instead, by specifying the ECHOItemIDs directly, users can get the metadata information through GetMetadata API transaction. The parameters that are required to perform the GetMetadata are ItemId, MessageFormat and PresentationDescription, which have been described in the Present API section. One restriction in using this transaction is that users have to make sure all of the item ids have the same data type, either COLLECION or GRANULE, otherwise an exception will be thrown to state which item id has a problem with the data type.

Code Example 64. GetMetadata API transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://partner-
test.echo.eosdis.nasa.gov:6600/echo/dtd/CatalogService.dtd">
<CatalogService>
  <GetMetadataRequest>
    <ItemID>C4380965-OPS_606AT2</ItemID>
    <PresentationDescription>
      <TupleType>
        <attributeName>Sensor</attributeName>
        <PrimitiveTypeName>
          <String></String>
        </PrimitiveTypeName>
      </TupleType>
      <TupleType>
        <attributeName>Campaign</attributeName>
        <PrimitiveTypeName>
          <String></String>
        </PrimitiveTypeName>
      </TupleType>
      <DTDType>
        <ECHO></ECHO>
      </DTDType>
    </PresentationDescription>
  </GetMetadataRequest>
</CatalogService>
```

Code Example 65. Get Metadata Response message.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE CatalogService SYSTEM "http://partner-test.echo.eosdis.nasa.gov:6600/echo/dtd/CatalogService.dtd">
<CatalogService>
  <GetMetadataResponse>
    <BooleanResult>
      <BooleanResultType><REQUEST_SUCCEEDED/></BooleanResultType>
    </BooleanResult>
    <ReturnData>
      <MessageFormat><XML/></MessageFormat>
      <payload><![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results SYSTEM "http://partner-test.echo.eosdis.nasa.gov:6600/echo/dtd/ECHOCollectionResults.dtd">
<results>
  <provider name='OPS_606AT2'>
    <result itemId='C4380965-OPS_606AT2'>
      <CollectionMetaData>
        <ECHOItemId>C4380965-OPS_606AT2</ECHOItemId>
        <DataSetId>GFCTestMODIS/Terra Raw Radiances in Counts 5-Min L1A Swath V042</DataSetId>
        <Platform>
          <PlatformShortName>Terra</PlatformShortName>
          <Instrument>
            <InstrumentShortName>MODIS</InstrumentShortName>
            <Sensor>
              <SensorShortName>MODIS</SensorShortName>
              <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
              <SensorTechnique>Radiometry</SensorTechnique>
            </Sensor>
          </Instrument>
        </Platform>
      </CollectionMetaData>
    </result>
  </provider>
</results>]]>
      </payload>
    </ReturnData>
  </GetMetadataResponse>
</CatalogService>
```

6.1.1.6 Query

The query must be specified as text under the <query> element in IIMSAQL query language and must conform to the IIMSAQLQueryLanguage.dtd in accordance with the IIMSAQL specification in the section on Alternative Query Language.

Note: The actual query (in IIMSAQL) is enclosed in the CDATA section in the QueryRequest message and hence a parser that is validating the QueryRequest message will not validate it. The query enclosed in the CDATA section should first be validated by itself against the IIMSAQLQueryLanguage.dtd before inserting it into the QueryRequest message.

The presentation portion is split into 2 areas. The first is specified by the ReturnType element that takes the values: RESULTS, RESULT_SET_ID, HITS, ITEM_IDS, NUMBER_OF_RESULTS. It also takes the value VALIDATE but it is not currently implemented and may be deprecated in the future. This is related to the kind of response required to the QueryRequest message.

RESULTS. Implies that the results should be returned as part of the response to the query request. When using this option, the rest of the details regarding presentation must be specified in the QueryRequest message. In addition, a key (ResultSetID) is also returned for subsequent retrievals of results in case all the results were not returned by the query response.

Code Example 66. Query Request transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
  <QueryRequest>
     <QueryExpression>
        <query>
   <![CDATA[
      …IIMSAQL query goes here…
   ]]>
       </query>
       <namespace>none</namespace>
       <QueryLanguage>
           <IIMSAQL/>
       </QueryLanguage>
    </QueryExpression>
    <ResultType>
       <RESULTS/>
    </ResultType>
    <IteratorSize>10</IteratorSize>
    <Cursor>1</Cursor>
    <PresentationDescription>
       <TupleType>
           <attributeName>Platform</attributeName>
           <PrimitiveTypeName><Integer/></PrimitiveTypeName>
       </TupleType>
       <TupleType>
           <attributeName>SensorShortName</attributeName>
           <PrimitiveTypeName><String/></PrimitiveTypeName>
       </TupleType>
       ... more TupleType specification ...
       <DTDType>
           <ECHO/>
       </DTDType>
    </PresentationDescription>
 </QueryRequest>
</CatalogService>
```

The query must be specified as text under the <query> element in IIMSAQL query language and must conform to the IIMSAQLQueryLanguage.dtd in accordance with the IIMSAQL specification in the section on Alternative Query Language.

Note: The actual query (in IIMSAQL) is enclosed in the CDATA section in the QueryRequest message and hence a parser that is validating the QueryRequest message will not validate it. The *query enclosed in the CDATA section should first be validated by itself against the IIMSAQLQueryLanguage.dtd before inserting it into the QueryRequest message.*

The presentation portion is split into 2 areas. The first is the response to the QueryRequest message. This is specified by the ResultType element that can take the following values:

RESULTS. Specifies that the results of the query should be returned as part of the response to the query request. When using this option, the rest of the details regarding presentation must be specified in the QueryRequest message. In addition, a key (ResultSetID) is also returned for subsequent retrievals of results in case all the results were not returned by the query response.

RESULT_SET_ID. Specifies that the response should return only a key (ResultSetID). A result set is generated, but no results are returned. Results must be subsequently retrieved using the Present transaction.

HITS. Specifies that the number of records in a result set be returned in addition to RESULT_SET_ID. A result set is generated but no results are returned. Results must be subsequently retrieved using the Present transaction.

ITEM_IDS. Specifies that the matched item ids and total number if item ids should be returned to the user directly. (Note: No result set ID is returned since results are not persisted in the system nor are they used to provide a stateless version of the Query transaction.) This implies that no RESULT is created within ECHO to store the results of the query. All the Ids of the granules/collections that satisfy the query are returned to the client. It is the client's responsibility to request the metadata for each individual granule/collection using the GetMetadata transaction discussed later.

COUNTS – Specifies that only the number of records in a result set be returned. The actual result set, however, is not created and stored within ECHO. Because there is no saving of the result set, this function will run faster than the other ResultTypes. This result type is recommended for queries only for the number of data matched.

Note: VALIDATE is an element that is not currently implemented and may be deprecated in the future.

The second area is applicable for QueryRequest message only if the results (part or all) are to be returned as part of the response to the QueryRequest message (by setting the ReturnType to RESULTS). This area is also used in the PresentRequest message; it is discussed in the Result Specification Section.

Code Example 67. Query Request message from "BOREAS" campaign.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<!-- Description:    Query ORNL data for granules from 'BOREAS' campaign. -->
<CatalogService>
   <QueryRequest>
      <QueryExpression>
         <query>
            <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!-- Search ORNL for granules from BOREAS campaign -->
<query>
   <for value="granules"/>
      <dataCenterId>
         <value>ORNL_TS1</value>
      </dataCenterId>
   <where>
      <granuleCondition>
```

```
         <CampaignShortName><value>'Boreas'</value></CampaignShortName>
      </granuleCondition>
   </where>
</query>
]]>
         </query>
         <namespace>none</namespace>
         <QueryLanguage>
           <IIMSAQL/>
         </QueryLanguage>
      </QueryExpression>
      <ResultType>
         <RESULT_SET_ID/>
      </ResultType>
   </QueryRequest>
</CatalogService>
```

Code Example 68. Query Request where NUMBER_OF_RESULTS are requested

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService SYSTEM "http://localhost:7001/echo/dtd/CatalogService.dtd">
<CatalogService>
          <QueryRequest>
                    <QueryExpression>
                              <query><![CDATA[

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query SYSTEM
             "http://localhost:7001/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
 <for value="granules"/>
  <dataCenterId>
     <value>GSFC-TEST</value>
  </dataCenterId>
 <where>
  <granuleCondition>
     <measuredParameters operator="AND">
       <measuredParameter>
        <measuredParameterName>'EV_1KM_RefSB'</measuredParameterName>
        <operationalQualityFlag>
            <value>'Passed'</value>
        </operationalQualityFlag>
        <QAPercentOutOfBoundsData>
            <range lower='30' upper='40'/>
        </QAPercentOutOfBoundsData>
       </measuredParameter>

       <measuredParameter>
        <measuredParameterName>'EV_1KM_RefSB'</measuredParameterName>
        <operationalQualityFlag>
```

```
            <value>'Passed'</value>
        </operationalQualityFlag>
        <QAPercentOutOfBoundsData>
            <range lower='39' upper='50'/>
        </QAPercentOutOfBoundsData>
       </measuredParameter>
     </measuredParameters>
  </granuleCondition>
 </where>
</query>
]]></query>
                              <namespace>none</namespace>
                              <QueryLanguage>
                                        <IIMSAQL/>
                              </QueryLanguage>
                    </QueryExpression>
                    <ResultType>
                              <NUMBER_OF_RESULTS/>
                    </ResultType>
          </QueryRequest>
</CatalogService>
```

6.1.1.7 PresentSavedQuery

This message is used to return the details of a query that has been saved. Guests do not have access to this transaction.

Code Example 69. PresentSavedQuery transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<!-- Description:  Lists saved result sets. -->
<CatalogService>
  <PresentSavedQueryRequest>
   <QueryName>mq1</QueryName>
</PresentSavedQueryRequest>
</CatalogService>
```

6.1.1.8 RemoveSavedQuery

This message is used to remove a saved query. This transaction is not available to Guests.

Code Example 70. RemoveSavedQuery transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
      <RemoveSavedQueryRequest>
         <QueryName>mq1</QueryName>
      </RemoveSavedQueryRequest>
</CatalogService>
```

6.1.1.9 RemoveSavedResultSet

This message is used to remove a saved result. This transaction is not available to Guests.

Code Example 71. RemoveSavedResultSet transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <RemoveSavedResultSetRequest>
      <ResultSetID>result1</ResultSetID>
   </RemoveSavedResultSetRequest>
</CatalogService>
```

6.1.1.10 SaveQuery

This is a request to save a query. Once a query has been saved, it may be executed by name rather than providing the entire query expression.

Code Example 72. SaveQuery transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <SaveQueryRequest>
   <QueryName>mq1</QueryName>
   <QueryExpression>
      <query>
      <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
 <for value="granules"/>
  <dataCenterId>
     <all/>
  </dataCenterId>
 <where>
  <granuleCondition>
     <onlineOnly/>
  </granuleCondition>
 </where>
</query>
]]>
      </query>
      <namespace>none</namespace>
      <QueryLanguage>
         <IIMSAQL></IIMSAQL>
      </QueryLanguage>
   </QueryExpression>
</SaveQueryRequest>
</CatalogService>
```

6.1.1.11 SaveResultSet

This message is used to persist a Result Set so that it may be retrieved or searched at a later time. This transaction is not available to Guests.

Code Example 73. SaveResultSet transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <SaveResultSetRequest>
      <ResultSetID>R1002_RS_978991090832</ResultSetID>
      <newResultSetID>result1</newResultSetID>
   </SaveResultSetRequest>
</CatalogService>
```

6.1.1.12 ListServicesForItem

This transaction lists the name and description of services available for the specified ECHO item.

Code Example 74. ListServicesforItemTransaction

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <ListServicesForItemRequest>
      <ECHOItemId>C395181-GSFC-TEST</ECHOItemId>
   </ListServicesForItemRequest>
</CatalogService>
```

6.1.1.13 PresentSearchOptions

This transaction requests a list of the settable options available for the Catalog Service. A user may request particular options by name; otherwise, all option definitions that can be set will be returned. The transaction is not currently implemented.

6.1.1.14 PresentSearchPreferences

The transaction is not currently implemented. This transaction will request a list of the user-selected and/or ECHO default preference settings for the Catalog Service.

6.1.1.15 UpdateSearchPreferences

The transaction is not currently implemented. This transaction will update the user's default search preferences. These preferences may be overridden in any given Query Request; however, setting these properties removes the necessity of providing this information as part of every Query Request. These preference values will be stored across sessions for registered users, but will expire at the end of a session for guest users.

6.1.2 Error Messages

The following table gives the possible error messages associated with each transaction within the Catalog Service. Some error messages are more complex then others and so will be treated in more detail in the following subsections.

Table 9: Catalog service error messages.

Transaction	Error Message	Description
IIMSAQL query specified is not well formed or does not conform to its DTD	See Section below.	See Section below.
Provider specified under the dataCenterId element does not exist	gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 10,column 26 Invalid data center ID [ORNL2] . DataCenter Id must be a valid ID NOT enclosed in quotes. Valid Ids are [ORNL,TEST_PROVIDER2,TEST_PROVIDER1])	The error message lists all the valid providers that can be specified as dataCenterIds. This list will vary according to the providers registered within ECHO at the time the query was issued.
Strings and text patterns specified for searches on campaign, dataSetId, spatialKeypwords etc., not enclosed in single quotes	See Section below.	See Section below.
Invalid date specified for any search element that requires dates like temporal, echoLastUpdate etc.	gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 18,column 47 Invalid Date)	Check whether the date is valid like for example date is trying to search on 29th of February.
Coordinates specified for spatial window points not within the spatial limits of the earth	gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 17,column 44 latitude value of IIMSPoint must be a number between [-90,90])	The error message will change whether the spatial window is being specified using Polygon, MultiPolygon etc., and also whether the latitude or longitude is out of range.
Cloud cover range specified not within 0-100	gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 16,column 38 Invalid range for element cloudCover. Attribute 'upper' of range element must be a number between [0,100])	Indicates that the Upper range specified is beyond 100.
Values specified for any ranges not conforming to upper>= lower	gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 16,column 38 Invalid range for element cloudCover.)	
Information related to presenting the results (when the request message specifies ResultType as RESULTS)	See Section below.	See Section below.

6.1.2.1 Query Error Messages

6.1.2.1.1 IIMSAQL query specified is not well formed or does not conform to its DTD

If the request message sent to ECHO for any transaction in the CatalogService is not well formed XML or does not conform to its DTD an attempt will be made to return the XML parser generated error message.

Error Message 3. Parser generated error when the "for" element is not specified in an IIMSAQL query.

```
gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in at line 16, column 9:The
content of element type "query" must match "(for,dataCenterId,where)".)
```

Error Message 4. Parser generated error when the IIMSAQL query fails to have correct matching tags

```
gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed
Error in  at line 14, column 41:The element type "value" must be terminated by the matching end-tag "</value>".)
```

But when the system attempts to return this error message, the error message itself invalidates the XML that is being returned to the user (because of the '<' and the '>' characters that are not escaped) and the ECHO system internally throws another error message. So the error message that the user will see will be the following.

Error Message 5. XML message generated by internal ECHO error.

```
<?xml version="1.0" standalone="no" ?>
<SessionManager>
   <Error>
      <![CDATA[
         Stopping after fatal error: The element type "Message" must be terminated by the matching end-tag
         "</Message>".
      ]]>
   </Error>
</SessionManager>
```

This is not the best error message to be sending to the user and future versions will correct this problem. In order to minimize the possibility of receiving such vague error messages, make sure the query in IIMSAQL is validated independently with its own DTD before embedding it in the CDATA section of the QueryRequest message.

6.1.2.1.2 Strings and text patterns specified for searches on campaign, dataSetId, spatialKeypwords etc., not enclosed in single quotes

There are two possible error messages – one for string values and one for text patterns.

Error Message 6. String value error message for searches not enclosed in single quotes.

```
gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 14,column 37
campaign values must be strings enclosed in single-quotes.)  or
```

Error Message 7. Text pattern error message for searches not enclosed in single quotes.

```
gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 14,column 49
textPattern element of campaign must be a string enclosed in single-quotes.)
```

The word "campaign" in the error message will be replaced by the search criteria that is violating this rule.

6.1.2.1.3 Spatial window specified as part of the spatial query is invalid

If a spatial window specified in the spatial query is invalid then an error message will be returned.

Error Message 8. Invalid spatial window error.

```
gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Invalid Window::Oracle error:13367)
```

This may not seem a very useful error message because it does not clearly state what about the spatial window is invalid. The Oracle error code 13367 is an indication of the problem, but the client would not be able to link this code to a useful error message. We will attempt to list the error codes and the corresponding explanation. This list may not be complete. If you receive an error code that is not listed, please let us know.

Table 10: Spatial error codes.

Oracle Spatial Error Code	Explanation of the error code
13349	The boundary of a polygon intersects itself. Correct the geometric definition of the spatial window
13351	Two or more rings of a complex polygon overlap. The inner or outer rings of a complex polygon overlap. All rings of a complex polygon must be disjoint. Correct the geometric description of the object.
13356	There are repeated points in the sequence of coordinates. Remove the redundant points.
13367	The orientation of the rings in the spatial window is incorrect. Make sure all the rings are in counter-clockwise order.

6.1.2.1.4 Information related to presenting the results (when the request message specifies ResultType as RESULTS)

The error messages in this case would be the same as those during Present transaction that are related to TupleTypes, IteratorSize and Cursor values. Please refer to the following section.

6.1.2.2 Query and Results Management Error Messages

6.1.2.2.1 PresentResults

Error Message 9. PresentResults error returned when user tries to present a Result Set that has not been created in the current session or previously saved.

```
gov.nasa.echo.services.ejb.catalogservice.UserResultNotFoundException(result=[ResultX]
for userId=[UserY] not found)
```

Where ResultX is the ResultSetID that was requested for presentation and UserY is the userId of the registered user or a temporary Id assigned to the Guest.

Error Message 10. PresentResults error returned when invalid TupleTypes are used.

```
Attributes [ShortN, DataSetID, ECHOUpdateDate] not valid for QueryType=[COLLECTIONS] and DTDType=[ECHO]
```

The DTDType by default is ECHO. The valid TupleTypes for the Present depend on the DTDType selected (current option is ECHO) and whether the results are for granules or collections. If the user specifies a TupleType that is not valid for this particular combination, the error message returned will include the list of invalid TupleTypes. For example, the error message shown above indicates the situation when trying to present results for Collections using the ECHO DTDType. We mistyped ShortName as ShortN, write DataSetID instead of DataSetId and write ECHOLastUpdate as ECHOUpdateDate.

6.1.2.2.2 Invalid values for either the Cursor or IteratorSize

If invalid values are specified for Cursor and Iterator then no error is generated, instead ECHO makes certain assumptions for the user:

1. In case the Cursor or Iterator values are negative numbers then the default results are returned (i.e. Cursor starting from 1 and Iterator size of 10)
2. In case the Cursor value is beyond the result set size, payload will contain the header of the results in XML but it will not actually have any results in it.

Error Message 11. Error returned when invalid values for either the Cursor or IteratorSize are used

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE results SYSTEM "http://api.echo.nasa.gov/echo/dtd/ECHOCollectionResults.dtd">
<results/>
```

In case the Iterator size takes the cursor beyond the total number of results, results are shown up to the last result in the result set.

6.1.2.2.3 SaveQuery

Error Message 12. SaveQuery - Error returned when a Guest is trying to save a query

```
Transaction available only to Registered Users
```

6.1.2.2.4 SaveResult

Error Message 13. SaveResult - Error returned when a Guest is trying to save a query

```
Transaction available only to Registered Users
```

Error Message 14. SaveResult - Error returned when a user tries to save a result with an existing name

```
Result with [savedName] name already exists
```

Error Message 15. SaveResult - Error returned when a user tries to save a result that does not exist yet

```
Result with name [resultSetId] does not exist for user [User101]
```

6.1.2.2.5 PresentSavedQuery

Error Message 16. PresentSavedQuery - Error returned when a Guest tries to present a saved query

```
Transaction available only to Registered Users
```

Error Message 17. PresentSavedQuery - Error returned when a user tries to present a saved query that does not exist yet

```
Query with [queryName] name does not exist
```

6.1.2.2.6 RemoveSavedQuery

Error Message 18. RemoveSavedQuery - Error returned when a Guest tries to remove a saved query

```
Transaction available only to Registered Users
```

Error Message 19. RemoveSavedQuery - Error returned when a user tries to remove a saved query that does not exist yet

```
Query with [queryName] name does not exist
```

6.1.2.2.7 Remove Saved ResultSet

Error Message 20. Remove Saved ResultSet - Error returned when a user tries to remove a result set that does not exist

```
Saved Result [myresult] for userId=User101 not found
```

6.1.2.2.8 List Saved Queries

Error Message 21. List Saved Queries - Error returned when a Guest tries to list saved queries

```
Transaction available only to Registered Users
```

6.1.2.2.9 List Saved ResultSets

Error Message 22. List Saved ResultSets - Error returned when a Guest tries to list saved result sets

```
Transaction available only to Registered Users
```

6.1.3 Alternative Query Language

ECHO Alternative Query Language (IIMSAQL) is a query language that defines the format for searches on collections (Discovery) and granules (Data or Inventory search) in the ECHO system. IIMSAQL is an XML based language. The DTD for IIMSAQL can be found at http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd

Code Example 75. General AQL structure.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
   <for value= "collections" or "granules" />
   <dataCenterId>
      <all/>
   </dataCenterId>
   <where>
      <list of conditions ... >
   </where>
</query>
```

Note: The <for/> tag shown in this example does not require a value attribute to be set, though it should be kept in mind that if you do not explicitly set this attribute it will default to "collections." If you do not set the value attribute your query will validate correctly, but an error can still be experienced if you later use granule settings for other attributes in your query.

The query language is designed to be data driven and declarative. The DTD defines the fields that can be queried as well as the specific data centers that should be used for the query. Under the <dataCenterId> tag, the user defines the data centers from which to search. Then for each field that can be queried, the user can specify the value(s) that must be satisfied by each returned result. The fields that can be searched on for collections occur as children of the <collectionCondition> element. The fields that can be searched on for granules occur as children of the <granuleCondition> element. Only those granules or collections that satisfy all the conditions form part of the result (i.e., ANDing of the conditions in the query is implied.). The list of conditions consists of <collectionCondition> elements for Discovery or <granuleCondition> elements for Inventory search.

The DTD for IIMSAQL specifies that the <where> tag can have any number of <collectionCondition> or <granuleCondition> elements. Moreover, it further specifies that each of these tags can contain any one of the search elements described below. However, note that for the query to be valid each of the search elements may appear ONLY once.

Code Example 76. Discovery search.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query><for value="collections"/>
   <dataCenterId><value>ORNL_TS1</value></dataCenterId>
   <where>
       <collectionCondition> … </collectionCondition>
       <collectionCondition> … </collectionCondition>
       …
   </where>
</query>
```

Code Example 77. Inventory search.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query><for value="granules"/>
   <dataCenterId><value>ORNL_TS1</value></dataCenterId>
   <where>
       <granuleCondition> … </granuleCondition>
       <granuleCondition> … </granuleCondition>
       …
   </where>
</query>
```

The data centers can be specified by explicitly listing them or by using the empty element <all/> to indicate that all the data centers should be searched. This is the default condition. i.e., if no condition is specified for <dataCenterId> element all data centers are searched. Do not enclose data center IDs in single-quotes.

Code Example 78. dataCenterID search element.

```
Search all data providers.

<dataCenterId><all/></dataCenterId>



Search for collections/granules in the LPDAAC_ECS or ORNL_TS1 data providers' metadata only.

<dataCenterId><list>
   <value>ORNL_TS1</value>
   <value>LPDAAC_ECS</value>
</list></dataCenterId>

Search for collections/granules in the ORNL data provider's metadata only.

<dataCenterId>
    <value>ORNL_TS1</value>
</dataCenterId>
```

6.1.3.1 Discovery Search

Searches for collections must enclose the search criteria in a <collectionCondition> element. If you want to search the negated criteria, you should set the negated attribute in the <collectionCondition> element to 'Y'. (e.g., The example below shows how to create a search for all collections except the ones with processing level 1 or 2).

Code Example 79. CollectionCondition element.

```
<collectionCondition negated='Y'>
  <processingLevel><list>
     <value>'1'</value>
     <value>'1A'</value>
     <value>'1B'</value>
     <value>'2'</value>
  </list></processingLevel>
</collectionCondition>
```

The negated attribute can be set to 'Y' for any criterion unless specified otherwise. For those criteria that do not support the negated version, even if the user specifies the attribute negated='Y' for the parent collectionCondition element, the criteria will remain positive.

The following criteria can be used to search for Collections.

Table 11: Discovery search criteria.

Search Criteria	XML Element
CampaignShortName	`<CampaignShortName>...</CampaignShortName>`
Dataset ID	`<dataSetId>...</dataSetId>`
ECHO Insert Date	`<ECHOInsertDate>...</ECHOInsertDate>`
ECHO Last Update	`<ECHOLastUpdate>...</ECHOLastUpdate>`
Online Collections Only	`<onlineOnly/>`
Parameter	`<parameter>...</parameter>`
Processing Level	`<processingLevel>...</processingLevel>`
Sensor Name	`<sensorName>...</sensorName>`
Short Name	`<shortName>...</shortName>`
Source Name	`<sourceName>...</sourceName>`
Spatial	`<spatial>...</spatial>`
Spatial Keywords	`<spatialKeywords>...</spatialKeywords>`
Temporal	`<temporal>...</temporal>`
Temporal Keywords	`<temporalKeywords>...</temporalKeywords>`
Additional Attribute Names	`<additionalAttributeNames>...</additionalAttributeNames>`
Orderable Items	`<orderable/>`
Version ID	`<versionId>...</versionId>`
Archive Center	`<archiveCenter>...</archiveCenter>`

6.1.3.2 Inventory Search

Searches for granules must enclose the search criteria in the <granuleCondition> element. As in the case of collectionCondition, if you want to search the negated criteria, set the negated attribute in the <granuleCondition> element to 'Y'. The negated attribute can be set to 'Y' for any criterion unless specified otherwise. For those criteria that do not support the negated version, even if the user specifies the attribute negated='Y' for the parent granuleCondition element, the criteria will remain positive.

Granules can be searched using the following criteria that are further discussed in the following sections.

Table 12: Inventory search criteria.

Search Criteria	XML Element
Only Granules with Browse Data	`<browseOnly/>`
CampaignShortName	`<CampaignShortName>...</CampaignShortName>`
Percentage of Cloud Cover	`<cloudCover>...</cloudCover>`
Dataset ID	`<dataSetId>...</dataSetId>`
ECHO Insert Date	`<ECHOInsertDate>...</ECHOInsertDate>`
ECHO Last Update	`<ECHOLastUpdate>...</ECHOLastUpdate>`
Either Day or Night Granules Only	`<dayNightFlag/>`
Only Global Granules	`<globalGranulesOnly/>`
Search on ECHO granule IDs (formerly granuleId)	`<ECHOGranuleID>...</ECHOGranuleID>`
Search on Granule UR (provider specific)	`<GranuleUR>...</GranuleUR>`
Online Granules Only	`<onlineOnly/>`
TwoDCoordinateSystem	`<TwoDCoordinateSystem>...</TwoDCoordinateSystem>`
ProducerGranuleID	`<ProducerGranuleID>...</ ProducerGranuleID>`
Additional Attributes	`<additionalAttributes>...</additionalAttributes>`
Sensor Name	`<sensorName>...<sensorName>`
Source Name	`<sourceName>...</sourceName>`
Spatial	`<spatial>...</spatial>`
Temporal	`<temporal>...</temporal>`
Orderable Items	`<orderable/>`
Version ID	`<versionId>...</versionId>`
PGE Name	`<PGEName>...</PGEName>`
PGE Version	`<PGEVersion>...</PGEVersion>`
Measured Parameters	`<measuredParameters>...</measuredParameters>`
Provider Production Date	`<providerProductionDate>…</providerProductionDate>`
Provider Insert Date	<providerInsertDate>…</providerInsertDate>

Effective with ECHO Release 7.0, LocalGranuleID search criteria was renamed ProducerGranuleID

6.1.3.3 Common Search Elements

6.1.3.3.1 CampaignShortName

Name(s) of the campaign or project that gathered data associated with the collection or granule. CampaignShortName can take a single-quoted string in a <value>, a list of single-quoted strings in a <list> or a textPattern in a <textPattern> element as defined in the section on Miscellaneous Elements.

Refer to the sub-section on textPattern under Miscellaneous Elements for detailed explanation on the pattern string. Later versions may support text patterns from Oracle InterMedia.

Code Example 80. CampaignShortName search element.

```
Search for collections/granules where campaignShortName = 'River'
<CampaignShortName>
   <value>'River'</value>
</CampaignShortName>


Search for collections/granules where campaignShortName = 'River' or  campaignShortName = 'Lake'
<CampaignShortName>
   <list>
      <value>'River'</value>
      <value>'Lake'</value>
   </list>
</CampaignShortName>


Search for collections/granules where campaignShortName contains the string 'AS_A%D'.
<CampaignShortName>
   <textPattern operator="LIKE">'AS_A%D'</textPattern>
</CampaignShortName>
```

6.1.3.3.2 DatasetID

This specifies the universal name of a collection. This element can be used to restrict the search to a few Collections. It can be specified using the <value> element to specify a single Collection, a <list> for a list of Collections or a <textPattern>. You can use the <value> or <list> only if you know the name of the Collection. This can be the case when you have performed a Discovery search and are now performing an Inventory search for granules within the collections of your interest.

Note: Beginning with ECHO release 5.5, the use of dataSetId is case sensitive. If the exact dataSetId is not used, searches could return incorrect results (including 0 hits).

Code Example 81. DatasetID search element.

```
Search for all collections whose dataSetId ends with '1A'
<collectionCondition>
   <dataSetId>
     <textPattern>'%1A'</textPattern>
   </dataSetId>
<collectionCondition>

Search for all granules whose dataSetId is 'LEAF CHEMISTRY, 1992-1993 (ACCP)' or 'ASTER DEM Product V002'only.
<granuleCondition>
   <dataSetId><list>
     <value>'LEAF CHEMISTRY, 1992-1993 (ACCP)'</value>
     <value>'ASTER DEM Product V002'</value>
   </list></dataSetId>
</granuleCondition>
```

6.1.3.3.3 OnlineOnly

Used to specify a search for granules or collections that are available online. Negated condition is not supported.

Code Example 82. OnlineOnly element.

```
<collectionCondition>
   <onlineOnly/>
<collectionCondition>
```

6.1.3.3.4 ECHO Insert Date

Used to specify a search based on when a collection or granule was inserted into ECHO. This can be specified only as date range. Negated condition is not supported.

Code Example 83. ECHOInsertDate element.

```
   <ECHOInsertDate>
       <dateRange>
         <startDate>
         <Date YYYY="2001" MM="04" DD="05"/>
         </startDate>
       </dateRange>
   </ECHOInsertDate>
```

6.1.3.3.5 ECHO Last Update

Used to specify a search based on when a collection or granule was last updated inside ECHO. This can be specified only as date range. Negated condition is not supported.

Code Example 84. ECHOLastUpdate element.

```
   <ECHOLastUpdate>
      <dateRange>
         <startDate>
           <Date YYYY="2001" MM="01" DD="01"/>
         </startDate>
         <stopDate>
           <Date YYYY="2001" MM="06" DD="01"/>
         </stopDate>
      </dateRange>
   </ECHOLastUpdate>
```

6.1.3.3.6 Sensor Name

This element is used to search for collections or granules based on the name of the sensor. Searching for a known value in a <value>, a list of known values in a <list>, or a text pattern in a <textPattern> is supported. A new search feature is added for searching with a list of values. Users can specify the relationship between each values which can be either "AND" or "OR". "AND" relationship means that results must match with every value specified. "OR" relationship means that results match with either one of the values specified. "OR" is the default setting for searching with multiple values.

Code Example 85. Sensor Name search element. This is an example specifying "AND" relationship searching with multiple values.

```
<granuleCondition>
    <sensorName operator="AND">
      <list>
        <value>'human observer'</value>
        <value>'rain gauge'</value>
      </list>
    </sensorName>
  </granuleCondition> >
```

6.1.3.3.7 Source Name

This element is used to search for collections or granules based on the name of the source. Searching for a known value in a <value>, a list of known values in a <list>, or a text pattern in a <textPattern> is supported. A new search feature is added for searching with a list of values. Users can specify the relationship between each value which can be either "AND" or "OR". "AND" relationship means that results must match with every value specified. "OR" relationship means that results match with either one of the values specified. "OR" is the default setting for searching with multiple values.

Code Example 86. Source Name search element. This is an example specifying "AND" relationship searching with multiple values.

```
<<granuleCondition>
    <sourceName operator="AND">
      <list>
        <value>'CV-580'</value>
        <value>'C-130'</value>
      </list>
    </sourceName>
  </granuleCondition>
```

6.1.3.3.8 Spatial

Spatial is used to restrict the search within a spatial extent of the earth. The negated condition is not supported. Also, specifying a spatial condition and globalGranulesOnly criterion in the same query is not allowed.

The spatial extent can be specified using the OpenGIS Consortium's (OGC) Geography Markup Language (GML) elements <Polygon>, <MultiPolygon> or <LString> found in GeometryCollection.dtd or by using the ECHO elements <IIMSPolygon>, <IIMSMultiPolygon>, or IIMSLine>. While using GML elements, the point MUST be specified with longitude first followed by the latitude.

A Polygon consists of one or more rings. Some rings can be embedded within others that define the interior of the polygon. The ECHO system can handle polygons with only one external ring and multiple internal rings.

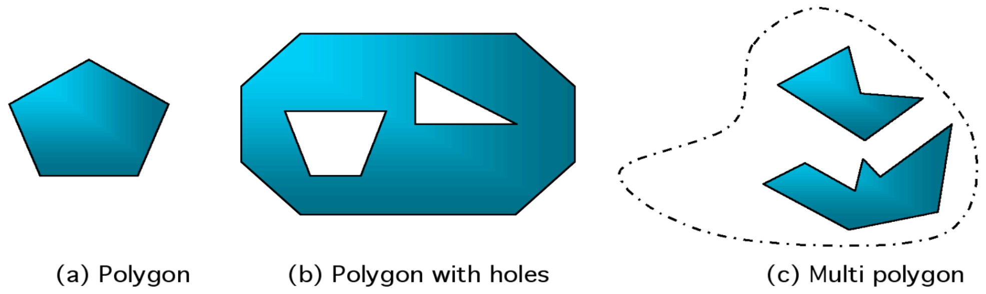


Figure 3. Allowed geometries for spatial data and query windows.

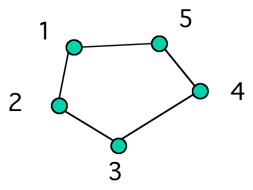


Figure 4. The points of a polygonal ring should be specified in counter clockwise order.

In order to define a ring, the last point in the ring should be the same as the first point in the ring. It is necessary to specify the points in each polygonal ring in counterclockwise order.

In order to specify multiple outer rings, the <MultiPolygon> or <IIMSMultiPolygon> elements should be used, with each outer ring in a separate <Polygon>. There should be no overlap between any of the outer rings defined.

<IIMSLine> or <LString> are used to specify a line on the Earth. It is necessary to specify the points in the line from one end point to the other.   <IIMSPoint> can be used to specify a single point on the Earth. Note that although <IIMSPoint> is used to describe corner points in a polygon or end points in a line, it is also describes a specific searchable geometry.

Spatial Operators available in the ECHO System are described below. Relationship between objects A and B is defined to be:

- **EQUAL.** They have the same boundary and interior.
- **TOUCH.** The boundaries intersect but the interiors do not.
- **WITHIN.** Interior and boundary of the first object is completely contained in the interior of the second.
- **CONTAINS.** The interior and boundary of the second object is completely contained in the interior of the first.
- **RELATE.** The objects are non-disjoint, i.e., their body and/or boundaries intersect.

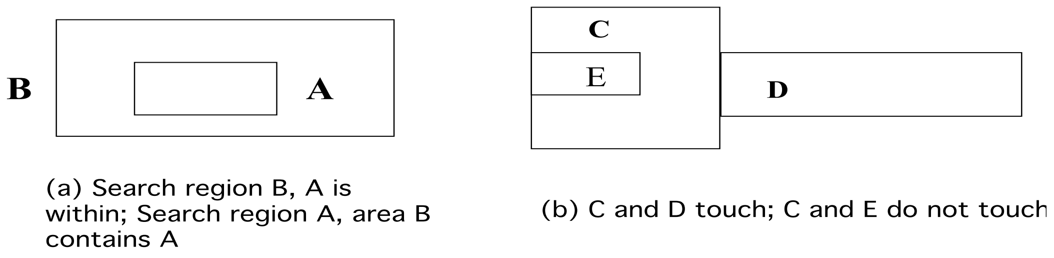

B
A
C
E
D
(a) Search region B, A is within; Search region A, area B contains A
(b) C and D touch; C and E do not touch

Figure 5. Types of spatial relationships.

The query is defined such that the user specified spatial extent, is the second object in the query. Hence use CONTAINS if you want to retrieve granules/collections that contain the specified polygon, use WITHIN if you want to retrieve granules/collections that are within the specified spatial extent and use RELATE to retrieve granules/collections that overlap in some way. RELATE is the default operator.

ECHO supports multiple spatial representations (or SpatialType). Spatial data of different spatial representation are described differently and hence are stored and queried differently. The spatial types that ECHO currently supports are described below.

NORMAL

Conventional spatial data are described as a certain shape such as a polygon, a multi-polygon, or a line. The spatial type for this type of data is categorized as "NORMAL". Spatial data of this type are stored in the database as an Oracle object to record its shape, spatial locations and coordinate system used, Cartesian or Geodetic.

Because the Oracle search for spatial data has different restrictions for different coordinate systems, some issues need to be noted. For the Cartesian system, Oracle allows the coverage to be as large as the whole earth but does not allow data that covers the pole or crosses the date line, so the search window must conform to these restrictions. For the Geodetic system, Oracle allows the data to cross the date line and the poles, but the total area of a single polygon cannot exceed half the earth, and the search window needs to conform to these constraints. Oracle does not validate the spatial search window before it executes the query. If the window does happen to be invalid no error is generated but the results produced may be incorrect. In order to reduce the effect of this issue on clients, ECHO validates the spatial window. Since the window validation needs to be irrespective of whether the data searched is Geodetic or Cartesian, the validation routine only validates the window to conform to the Geodetic coordinate system. It is the client's responsibility to make sure that they conform to the extra restriction imposed when querying the Cartesian system data. In addition, even though the Cartesian system allows the query window to be as large as the whole earth, ECHO will restrict the window to be no larger than half the area of the earth. Smaller sized spatial search windows result in faster response times.

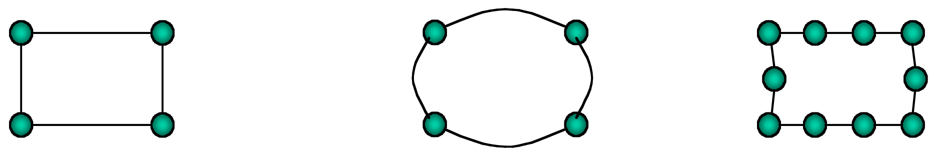

Polygon in Cartesian system connected by straight lines
Polygon in Geodetic system connected by great circle arcs
Same Polygon specified in Geodetic system specified with more points to more accurately represent the real image extent

Figure 6. Polygon area defined depends on the number of points and the coordinate system used.

The actual area on the earth that is queried depends on the coordinate system used by the provider and the number of points in the query window. When searching data from providers using Cartesian coordinates the search polygon is converted to a Cartesian polygon with points connected using straight lines, The search polygon is converted to a Geodetic polygon while searching data from providers using the Geodetic coordinate system and the points are connected using great circle arcs. This may result in slightly different results for data from providers using Cartesian and those using the Geodetic coordinate system. To ensure better accuracy, represent the search window with more than just the corner points.

Oracle Spatial search is a two-step process. The first step filters the data by querying the spatial window against the bounding box of the spatial data. The second step is a slower more accurate polygonal geometry search for exact results on the results from the first step. A small window results in more data being eliminated by the first filter, so the search will run faster.

Skewed polygonal windows or data will have lots of wasted space in the bounding box will result in more false positives passing through the fast filter.

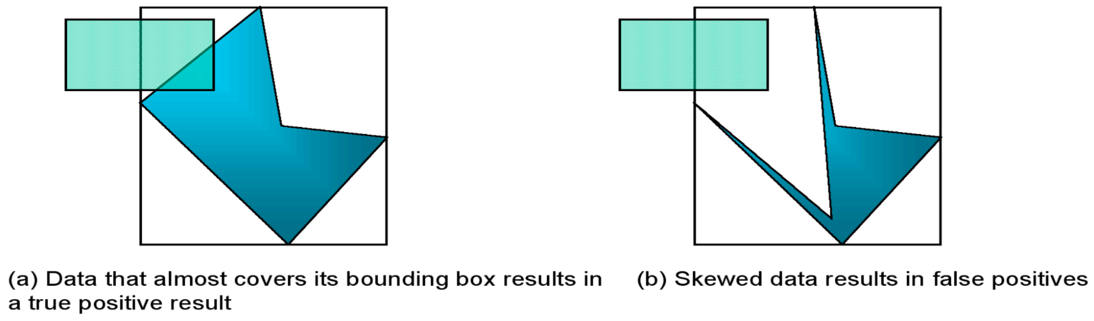

(a) Data that almost covers its bounding box results in a true positive result
(b) Skewed data results in false positives

Figure 7. Data orientation with respect to its bounding rectangle may play a role in the response time of a spatial query.

GLOBAL

A special case is if the spatial coverage of the data is the entire earth. The data does not come with any specific spatial value, but rather a flag to denote its spatial as global. Currently ECHO only supports global spatial searches on granules.

ORBIT

Spatial data are orbit-based and generated from satellites running along orbits. Because an orbit travels at a fixed direction with a fixed declination angle and the covered area is a stripe of fixed width, the spatial coverage of a granule can be derived from the latitude where the granule starts and ends and the crossing longitude on the equator where its orbit originally starts. Thus, the orbit-based spatial data are not stored as shape object but as the original data. ECHO uses an algorithm called "backtrack" to perform searches on orbit-based data. Spatial coverage can be as large as the entire earth surface including polar area. ECHO does simple validation against the spatial window to make sure it does not exceed the scope of the earth (the latitude is within the range –90 to 90 and the longitude is within the range –180 to 180). There are some limitations in the current implementation. First, ECHO only supports searching on single-orbit granules. Secondly, the only supported spatial operator is RELATE.

<SpatialType> tag is used to specify the spatial type and is an optional field in AQL. If the user opts not to specify this field, ECHO performs a default search. By default, ECHO performs search on NORMAL (Cartesian and Geodetic), CONSTANT_COVERAGE and ORBIT. GLOBAL granules are not returned in the default case.

Code Example 87. Spatial Search Element

```
<spatial operator="RELATE">
  <Polygon>
    <LRing>
      <CList>-120, -30, -100, -60, 5, -90,
      -120, -60, 160, 5, 160, 60, 120, 85,
      5, 85, -120, 30, -120, -30</CList>
    </LRing>
    <LRing>
      <CList>80, 20, 80, 60, 20, 60,
             20, 20, 80, 20</CList>
    </LRing>
  </Polygon>
  <SpatialType>
     <list>
      <value>NORMAL</value>
      <value>ORBIT</value>
      <value>GLOBAL</value>
    </list>
  </SpatialType>
</spatial>
```

Code Example 88. A multi-polygon spatial query.

```
<spatial operator='RELATE'>
   <IIMSMultiPolygon>
      <IIMSPolygon>
         <IIMSLRing>
            <IIMSPoint lat="85" long="-50"/>
            <IIMSPoint lat="70" long="-60"/>
            <IIMSPoint lat="60" long="-50"/>
            <IIMSPoint lat="70" long="-40"/>
            <IIMSPoint lat="85" long="-50"/>
         </IIMSLRing>
      </IIMSPolygon>
      <IIMSPolygon>
         <IIMSLRing>
            <IIMSPoint lat="-85" long="-50"/>
            <IIMSPoint lat="-70" long="-40"/>
            <IIMSPoint lat="-60" long="-50"/>
            <IIMSPoint lat="-70" long="-60"/>
            <IIMSPoint lat="-85" long="-50"/>
         </IIMSLRing>
      </IIMSPolygon>
   </IIMSMultiPolygon>
</spatial>
```

Code Example 89. A single point spatial query.

```
<spatial operator='RELATE'>
   <IIMSPoint long='-75.0' lat='35.0'/>
< spatial>
```

Code Example 90. A spatial query using a line defined by GML elements.

```
<spatial operator='EQUAL'>
  <LString>
    <CList>70, 80, 80, 80, 90, 90</CList>
  </LString>
</spatial>
```

6.1.3.3.9 Temporal

Used to specify the temporal extent of the data. The dates are stored as Gregorian dates, and the times are stored in GMT. This search allows only dates in YYYY-DD-MM format. A range of time as beginning and ending dates can be specified or a periodic temporal range can be specified as start date, end date and the start and end day of each year can be specified. The start and end day should be an integer between 1 and 365 with start day <= end day. The end day can be specified as 366, but if the temporal range contains a non-leap year then an error will be generated.

Code Example 91. Temporal search element.

```
Searches for granule/collections whose temporal range overlaps with the time between May 12, 1990 to June 13, 1992.
<temporal>
   <startDate><Date YYYY="1990" MM="5" DD="12"/></startDate>
   <stopDate><Date YYYY="1992" MM="6" DD="13"/></stopDate>
</temporal>


Searches for granules/collections whose temporal range overlaps with the time ranges: May 16, 1990 - July 14, 1990 or
January 5, 1991 - March 5, 1991 or January 5, 1991 - February 10, 1991.
<temporal>
   <startDate><Date YYYY="1990" MM="5" DD="12"/></startDate>
   <stopDate><Date YYYY="1992" MM="2" DD="10"/></stopDate>
   <startDay value="5"/>
   <endDay value="60"/>
</temporal>
```

6.1.3.3.10 Orderable Items

Used to specify search for granules or collections that can be ordered.

Code Example 92. Orderable element.

```
<collectionCondition>
   <orderable/>
<collectionCondition>
```

6.1.3.3.11 Version ID

This element is used to search for collections or granules based on the version identifier of the data collection. Searching for a known value in a <value>, a list of known values in a <list>, or a text pattern in a <textPattern> is supported.

Code Example 93. Version ID search element.

```
<collectionCondition>
     <versionId>
         <value>'0'</value>
     </versionId>
</collectionCondition>

<granuleCondition>
   <versionId>
       <textPattern operator="LIKE">'3.%'</textPattern>
   </versionId>
</granuleCondition>
```

6.1.3.4 Search Elements Specific to Discovery

6.1.3.4.1 Parameter

Specify the Geophysical terms associated with a collection. This search term can be used only for collection searches not for granule searches. If a list or a single value is specified, then the search looks for an exact match between the string specified and the data stored. Hence if the search value is 'Imagery' and the data stored has the value 'Satellite Imagery' the data is not returned since there was not exact match. Hence it is more useful to use <textPattern> element for the search.

Code Example 94. Parameter search element.

```
<parameter>
   <textPattern operator="LIKE">'%Imagery%'</textPattern>
</parameter>
```

6.1.3.4.2 Processing Level

Level to which the Data (collection/granule) has been processed. Example values are 0, 1, 1A, 1B, 2, 3, 4. The search can specify one or more processing levels. The values must be single-quoted strings.

Code Example 95. Processing Level search element. Search for all collections at processing level 1 or 1A or 1B.

```
<collectionCondition>
   <processingLevel>
      <list>
         <value>'1'</value>
         <value>'1A'</value>
         <value>'1B'</value>
      </list>
   </processingLevel>
</collectionCondition>
```

Code Example 96. Processing Level search element. Search for all collections at processing level 1, or 1A, or 1B and, 1C etc if they exist. This also returns results for collections with processing levels 13, 14, etc.

```
<collectionCondition>
   <processingLevel>
      <textPattern
        operator="LIKE">'1_'</textPattern>
   </processingLevel>
</collectionCondition>
```

6.1.3.4.3 Additional Attribute Name

This element is used to search for collections based on the names of the additional attributes in the collections. Searching for a known value in a <value>, a list of known values in a <list>, or a text pattern in a <textPattern> is supported.

Code Example 97. Additional Attribute Name search element

```
<collectionCondition>
   <additionalAttributeName>
      <list>
         <value>'value_1'</value>
```

```
        <value>'value_3'</value>
    </list>
  </additionalAttributeName
</collectionCondition>
```

6.1.3.4.4 Short Name

Short name of a collection. May or may not be unique for a particular provider.

Code Example 98. Short Name search element. Search for all collections that have shortNames starting with 'BOREAS.'

```
<collectionCondition>
  <shortName>
    <textPattern>'BOREAS%'</textPattern>
  </shortName>
</collectionCondition>
```

6.1.3.4.5 Spatial Keywords

Specifies a word or phrase that summarizes the spatial region covered by the collection. A collection may cover several regions.

Code Example 99. Spatial Keywords search element – Search for all collections that cover Africa or Bermuda or the Indian Ocean.

```
<collectionCondition>
  <spatialKeywords>
    <list>
      <value>'Africa'</value>
      <value>'Bermuda'</value>
      <value>'Indian Ocean'</value>
    </list>
  </spatialKeywords>
</collectionCondition>
```

Code Example 100. Spatial keywords search element – Search for all collections that cover some regions of the Americas.

```
<collectionCondition>
  <spatialKeywords>
    <textPattern  operator="LIKE">'%america%'</textPattern>
  </spatialKeywords>
</collectionCondition>
```

6.1.3.4.6 Temporal Keywords

Specifies a word or phrase that summarizes the temporal characteristics referenced in the collection.

Code Example 101. Temporal keywords search element – Search for all collections that have data during the spring or fall.

```
<collectionCondition>
  <temporalKeywords>
```

```
      <list>
         <value>'spring'</value>
         <value>'fall'</value>
      </list>
   </temporalKeywords>
</collectionCondition>
```

6.1.3.4.7 Archive Center

This element is used to search for collections based on the center where colleciton is archived. Searching for a known value in a <value>, a list of known values in a <list>, or a text pattern in a <textPattern> is supported.

Code Example 102. Archive Center search element.

```
  <collectionCondition>
     <archiveCenter>
       <value>'ORNL DAAC'</value>
     </archiveCenter>
  </collectionCondition>
```

6.1.3.5 Search Elements Specific to Inventory

6.1.3.5.1 Only Granules with Browse Data

Used to specify that only granules that have browse data available should be returned. Negated condition is not supported.

Code Example 103. Only Granules with browse data search element.

```
<granuleCondition><browseOnly/>
</granuleCondition>

or

<granuleCondition><browseOnly value="Y"/>
</granuleCondition>
```

6.1.3.5.2 Percentage of Cloud Cover

Range of percentage of scene cloud coverage in a granule. The value in each attribute of range should be a float between [0,100] that specifies the range in percentage of cloud coverage in a granule that is returned. Negated condition is not supported.

Code Example 104. Percentage of Cloud Cover search element. Return only those granules that have cloud cover between 10% and 20%.

```
<granuleCondition>
   <cloudCover>
     <range lower="10" upper="20"/>
   </cloudCover>
</granuleCondition>
```

6.1.3.5.3 Day, Night, or Both Granules Only

Specifies that granules that are gathered during daylight or nighttime or both be returned. If this criterion is not specified then granules gathered at any and all times of day are returned. Negated condition is not supported.

Code Example 105. Day, Night, or Both Granules Only search element. Select granules gathered at least partly during daylight.

```
<granuleCondition>
   <dayNightFlag value="DAY"/>
</granuleCondition>
```

Code Example 106. Day, Night, or Both Granules Only search element. Select granules gathered at least partly during the night.

```
<granuleCondition>
   <dayNightFlag value="NIGHT"/>
</granuleCondition>
```

Code Example 107. Day, Night, or Both Granules Only search element. Select granules gathered partly in night and partly during the daylight.

```
<granuleCondition>
   <dayNightFlag value="Both"/>
</granuleCondition>
```

6.1.3.5.4 Only Global Granules

Search only for Global granules (granules that span the whole earth). Do not specify both spatial criteria and globalGranulesOnly as granule conditions. Only one of the criteria is allowed. Negated condition is not supported.

Code Example 108. Only Global Granules search element.

```
<granuleCondition>
   <globalGranulesOnly/>
</granuleCondition>

or

<granuleCondition>
   <globalGranulesOnly value="Y"/>
</granuleCondition>
```

6.1.3.5.5 Search on ECHO Granule IDs

Specify an inventory search for specific granules. This search does not need to specify any spatial or temporal constraint. This can be used as a second stage query when the user knows the list of granules he is interested in from a previous query. The search is then limited to granules in the list. The list may contain granules from different data sets. The search is performed on ECHO_GRANULE_ID; an ECHO wide unique ID. Negated condition is not supported. The names of the granules must be enclosed in single quotes.

Code Example 109. Search on ECHO Granule IDs search element.

```
<granuleCondition>
   <ECHOGranuleID>
      <list>
         <value>'G1984-ORNL'</value>
      </list>
   </ECHOGranuleID>
</granuleCondition>
```

6.1.3.5.6 Search on Granule UR

Specify an inventory search for specific granules. Like granule IDs, this criterion can be used as a secondary stage query. Instead of using the ECHO specific id for the granules, this criterion allows the user to use the provider given name for the granules. Provider given names are more meaningful and easier to remember. Since the granule names given by the provider are not guaranteed to be unique within ECHO, if a granule with the same name exists for more than one dataCenterId specified in the query, all will be returned in the results. Negated condition is not supported. The names of the granules must be enclosed in single quotes.

Code Example 110. Search on Granule UR search element.

```
<granuleCondition>
   <GranuleUR>
      <list>
         <value>'ACCP_CANOPYCHEM.df_can_calc.dat'</value>
         <value>'ANGLIA_10YRCLIMATE.cfrs1120.zip'</value>
      </list>
   </GranuleUR>
</granuleCondition>
```

6.1.3.5.7 Search by Two Dimensional Coordinate System

This is an alternative to specifying a spatial or temporal range. For some collections, granules can be located by searching a range of values on a two dimensional grid. For example: Path/Row for Landsat and X/Y for Modis Grid data. A negated condition is not supported. (i.e., if the parent element has the attribute negated='y' the NOT operator is not added and the condition remains positive).

The TwoDCoordinateSystem criterion needs three elements: the TwoDCoordinateSystemName, the Coordinate1 range and the Coordinate2 range. The Coordinate1 and Coordinate2 may also take a single value. It returns all granules that are specified with the user specified TwoDCoordinateSystemName and overlap with the Coordinate1 and Coordinate2 range specified. Coordinate1 range and Coordinate2 range must have both lower and upper attributes specified.

Code Example 111. TwoDCoordinateSystem search element .

```
Range value example:

<granuleCondition>
    <TwoDCoordinateSystem>
        <TwoDCoordinateSystemName><value>'WRS2'</value></ TwoDCoordinateSystemName>
        <Coordinate1><range lower='15' upper='20'/></Coordinate1>
        <Coordinate2><range lower='33' upper='42'/></Coordinate2>
    </ TwoDCoordinateSystem>
 </granuleCondition>

Single value example:
```

```
<granuleCondition>
    < TwoDCoordinateSystem>
       <TwoDCoordinateSystemName><value>'WRS2'</value></TwoDCoordinateSystemName>
       <coordinate1><value>15<value/></coordinate1>
       <coordinate2><range lower='33' upper='42'/></coordinate2>
    </ TwoDCoordinateSystem >
</granuleCondition>
```

6.1.3.5.8 ProducerGranuleID

Specify an inventory search for specific granules. This represents the ID assigned to this data by the Science team that produced this data. Instead of using the ECHO specific IDs for the granules, this criterion allows the user to use the producer's given name for the granules. Granule names are available to the Science teams and are easier to search on. Since the granule names given by the producer are not guaranteed to be unique within ECHO, if a granule with the same name exists for more than one dataCenterId specified in the query, all will be returned in the results. Negated condition is not supported. For example, if the parent granuleCondition element has the attribute negated ='y' the NOT operator is not added and the condition remains positive. The names of the granules must be enclosed in single quotes.

Code Example 112. ProducerGranuleID search element.

```
<granuleCondition>
   <ProducerGranuleID>
      <list>
         <value>'MOD03.A2000107.0930.003.2002056052717.hdf'</value>
         <value>'MOD01.A2000107.0740.003.2002056051321.hdf'</value>
      </list>
   </ProducerGranuleID>
</granuleCondition>
```

6.1.3.5.9 Additional Attributes

This element is used to search for granules based on one or many groups of the additional attributes by specifying the additional attribute name/value pair for exact match. The function includes searching on multiple additional attributes. Users can specify the relationship between groups which can be either "AND" or "OR". "AND" relationship means that result granules must match across every group. "OR" relationship means that result granules match with either one of the groups. "OR" is the default setting. Within each additional attribute, a <additionalAttributeName> must be specified in single quote for exact match, the <additionalAttributeValue> can be <value>, <list>, <textPattern>, <range>, or <dateRange>.

Code Example 113. Additional Attributes search element. This is an example specifying "AND" relationship between two additional attribute groups.

```
<granuleCondition>
     <additionalAttributes operator='AND'>
     <additionalAttribute>
        <additionalAttributeName>'AveragedFocalPlane1Temperature'</additionalAttributeName>
        <additionalAttributeValue>
            <range lower='169' upper='170'/>
        </additionalAttributeValue>
      </additionalAttribute>

      <additionalAttribute>
```

```
        <additionalAttributeName>'AveragedFocalPlane2Temperature'</additionalAttributeName>
        <additionalAttributeValue>
            <range lower='269' upper='270'/>
        </additionalAttributeValue>
     </additionalAttribute>
    </additionalAttributes>
  </granuleCondition>
```

Code Example 114. Additional Attributes search element. This is an example specifying "OR" relationship between two additional attribute groups.

```
<granuleCondition>
     <additionalAttributes operator='OR'>
     <additionalAttribute>
        <additionalAttributeName>'AveragedFocalPlane1Temperature'</additionalAttributeName>
        <additionalAttributeValue>
            <range lower='169' upper='170'/>
        </additionalAttributeValue>
      </additionalAttribute>

      <additionalAttribute>
        <additionalAttributeName>'AveragedFocalPlane2Temperature'</additionalAttributeName>
        <additionalAttributeValue>
            <range lower='269' upper='270'/>
        </additionalAttributeValue>
     </additionalAttribute>
    </additionalAttributes>
  </granuleCondition>
```

6.1.3.5.10 PGE Name

This element is used to search for granules based on the name of the product generation executive (PGE). Searching for a known value in a <value>, a list of known values in a <list>, or a text pattern in a <textPattern> is supported.

Code Example 115. PGE Name search element.

```
  <granuleCondition>
     <PGEName>
       <value>'test_PGEName'</value>
     </PGEName>
  </granuleCondition>
```

6.1.3.5.11 PGE Version

This element is used to search for granules based on the version of PGE that originally produced the granule. Searching for a known value in a <value>, a list of known values in a <list>, or a text pattern in a <textPattern> is supported.

Code Example 116. PGE Version search element.

```
  <granuleCondition>
     <PGEVersion>
```

```
      <value>'3.0.0'</value>
    </PGEVersion>
</granuleCondition>
```

6.1.3.5.12 Collection Short Name

This element is used to search for granules based on the short name of the collection it belongs to. Searching for a known value in a <value>, a list of known values in a <list>, or a text pattern in a <textPattern> is supported.

Code Example 117. Collection Short Name search element.

```
<granuleCondition>
    <collectionShortName>
      <value>'MOD03'</value>
    </collectionShortName>
</granuleCondition>
```

6.1.3.5.13 Measured Parameters

This element is used to search for granules based on one or many groups of measured science parameters. Users can specify the relationship between groups which can be either "AND" or "OR". "AND" relationship means that result granules must match across all groups. "OR" relationship means that result granules can match with any of the groups. "OR" is the default setting. Within each measured parameter, a <measuredParameterName> must be specified in single quotes for exact matching. The detailed measured parameters can be <value>, <list>, and <textPattern> type, or <range> and <dateRange> type.

Code Example 118. Measured Parameters search element. This is an example specifying "AND" relationship between two measured parameter groups.

```
<granuleCondition>
    <measuredParameters operator="AND">
      <measuredParameter>
       <measuredParameterName>'EV_1KM_RefSB'</measuredParameterName>
       <operationalQualityFlag>
           <value>'Passed'</value>
       </operationalQualityFlag>
       <QAPercentOutOfBoundsData>
           <range lower='30' upper='40'/>
       </QAPercentOutOfBoundsData>
      </measuredParameter>

      <measuredParameter>
       <measuredParameterName>'EV_1KM_RefSB'</measuredParameterName>
       <operationalQualityFlag>
           <value>'Passed'</value>
       </operationalQualityFlag>
       <QAPercentOutOfBoundsData>
           <range lower='39' upper='50'/>
       </QAPercentOutOfBoundsData>
      </measuredParameter>
    </measuredParameters>
</granuleCondition>
```

Code Example 119. Measured Parameters search element. This is an example specifying "OR" relationship between two measured parameter groups.

```
<granuleCondition>
   <measuredParameters operator="OR">
     <measuredParameter>
      <measuredParameterName>'EV_1KM_RefSB'</measuredParameterName>
      <operationalQualityFlag>
          <value>'Passed'</value>
      </operationalQualityFlag>
      <QAPercentOutOfBoundsData>
          <range lower='30' upper='40'/>
      </QAPercentOutOfBoundsData>
     </measuredParameter>

     <measuredParameter>
      <measuredParameterName>'EV_1KM_RefSB'</measuredParameterName>
      <operationalQualityFlag>
          <value>'Passed'</value>
      </operationalQualityFlag>
      <QAPercentOutOfBoundsData>
          <range lower='39' upper='50'/>
      </QAPercentOutOfBoundsData>
     </measuredParameter>
   </measuredParameters>
</granuleCondition>
```

6.1.3.5.14 ProviderInsertDate

This feature allows the user to search on granules that are within a desired date-time range of ProviderInsertDate. StartDate is a required field, and StopDate is optional. The Date format is:

<Date YYYY="2001" MM="01" DD="01" HH="00" MI="00" SS="00"/>

Code Example 120. Search on ProviderInsertDate

```
<granuleCondition>
   <providerInsertDate>
     <dateRange>
      <startDate>
          <Date YYYY="2001" MM="01" DD="01" HH="00" MI="00" SS="00"/>
      </startDate>
      <stopDate>
          <Date YYYY="2003" MM="12" DD="01" HH="23" MI="59" SS="59"/>
      </stopDate>
     </dateRange>
   </providerInsertDate>
</granuleCondition>
```

6.1.3.5.15 ProviderProductionDate

This feature allows the user to search on Granules that are within a desired date-time range of ProviderProductionDate. StartDate is a required field, and StopDate is optional. The Date format is: <Date YYYY="2001" MM="01" DD="01" HH="00" MI="00" SS="00"/>

Code Example 121. Search on ProviderProductionDate

```
<granuleCondition>
   <providerProductionDate>
     <dateRange>
      <startDate>
          <Date YYYY="2001" MM="01" DD="01" HH="00" MI="00" SS="00"/>
      </startDate>
      <stopDate>
          <Date YYYY="2003" MM="12" DD="01" HH="23" MI="59" SS="59"/>
      </stopDate>
     </dateRange>
   </providerProductionDate>
</granuleCondition>
```

6.1.3.6 Miscellaneous Elements

6.1.3.6.1 list

This element is used to specify a list of values. The values may be numbers, dates or strings. Each individual value must appear as text of the value element. If a list of strings is specified it must be a single-quoted string with no wild-card characters. The parent element will be searched using Oracle's IN operator for an exact match with any value in the list. The search is case insensitive.

In general, interpretation of the list will be dependent on the parent element in which it appears. Refer to the specific parent element in which the list appears for a detailed explanation.

Code Example 122. List element.

```
<list>
   <value>'Africa'</value>
   <value>'Bermuda'</value>
   <value>'Indian Ocean'</value>
</list>
```

6.1.3.6.2 textPattern

The textPattern element uses the Oracle operator "LIKE" to perform pattern matching. Because of this pattern matching, exact specification of what is desired is not needed. The code example below shows how to invoke the pattern matching. Note that the operator must be specified. The string specified to be matched must be a single-quoted string with or without wild-card characters.

The wild characters supported are '%' and '_'. '%' is a placeholder for replace by any number of characters. "_" is a placeholder for replace any one character.

Table 13: Wild card use.

Oracle Pattern for the LIKE operator	Interpretation
`'%JOHN%'`	Strings containing 'JOHN' as a sub-string
`'MARY%'`	Strings beginning with the characters 'MARY'
`'%BOB'`	Strings ending in 'BOB'
`'D_VE'`	match words like 'DAVE', 'DOVE', etc.

You can include the actual characters "%" or "_" in the pattern by using the ESCAPE character '\'. If the escape character appears in the pattern before the character "%" or "_" then Oracle interprets this character literally in the pattern, rather than as a special pattern matching character.

Table 14: Escape character use.

Oracle Pattern for the LIKE operator with the ESCAPE character '\'	Interpretation
`'JOHN\%THAN'`	String 'JOHN%THAN'
`'JOHN%THAN'`	Strings 'JOHNNATHAN', 'JOHNASDTHAN' etc.
`'JOHN\\MARY'`	String 'JOHN\MARY'

If the parent element (<collectionCondition> or <granuleCondition>) has the attribute negated with value 'Y' then the search will look for a string NOT like the pattern specified. The search is case insensitive.

Code Example 123. TextPattern element.

```
<spatialKeywords>
  <textPattern  operator="LIKE">'america%'</textPattern>
</spatialKeywords>
```

6.1.3.6.3 value

The value element is used to specify one value. Refer to each individual parent element for a detailed explanation. In general, if the element represents a String pattern the search will be case insensitive.

Code Example 124. Value search element.

```
<value>12.6</value>

or

<value>'Asx'</value>
```

6.1.3.6.4 Date

The date element is used to specify a date value.

Code Example 125. Date search element.

```
<Date YYYY="2000" MM="05" DD="03"/>

or

<Date YYYY="2000" MM="10" DD="09" HH="21" MI="04" SS="32"/>
```

6.1.3.6.5 dateRange

The dateRange element is used to specify a range of time. It can include just a startDate, just a stopDate or both. If only a startDate is declared, it only searches from that time forward with no stop date. If only a stopDate is declared, it only searches from the time up until the stop date.. If it includes both a startDate and a stopDate then it is declaring any time between those two dates only, for example, between January 1, 2005 and January 31, 2005.

Code Example 126. DateRange search element.

```
<dateRange>
   <startDate><Date YYYY="2000" MM="05" DD="03"/></startDate>
   <stopDate><Date YYYY="2000" MM="10" DD="09"/></stopDate>
</dateRange>

or

<dateRange>
   <stopDate>
      <Date YYYY="2000" MM="10" DD="09" HH="04" MI="12" SS="34"/>
   </stopDate>
</dateRange>
```

6.1.4 Visibility of Results

When a user executes a Query transaction, the query is applied to all the data in ECHO to create the result set. But when the results are to be presented, not all results may be visible to the user. The visibility of the data depends on the rules defined by the Provider of the data and the privileges that the user has been granted. Please refer to ECHO Requirements on Visibility for more details.

If a particular result is not visible to the user, the result will specify the ECHO id for that result and specify that it is not visible. The user may later inquire with ECHO Ops as to what steps should be taken in order for the results to be visible.

Code Example 127. Results not visible to the user.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results SYSTEM "http://api.echo.nasa.gov/echo/dtd/ECHOGranuleResults.dtd">
<results>
  <provider name='ORNL_TS1'>
    <result number='1'>G70958-ORNL_TS1 is not visible</result>
    <result number='2'>G70959-ORNL_TS1 is not visible</result>
  </provider>
</results>
```

6.1.5 Deleted Results

When a user executes a Query, all the data that conforms to the Query are saved as the result set of the Query. This result set is saved within ECHO. The user can come back anytime later to view this result set (provided it has not already been deleted from ECHO). It is possible that between the time the Query is executed and the results are presented, some of the data might be deleted from ECHO due to a request from the Provider. In that case the result will specify the ECHO id for that result and specify it as deleted.

The DTD for the result (as noted in the XML itself) is located at:

http://api.echo.nasa.gov/echo/dtd/ECHOGranuleResults.dtd

Every Result XML file begins with the root element, <results>. The results are grouped by data provider. Hence, the next element under the result element is the provider element with an attribute name that specifies the name of the provider.

6.1.6 Sample Queries

Please note that some of the sample queries may result in 0 hits.

Code Example 128. Queries for Collections (Discovery Search)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for collections from ORNL that have
  parameter value that contains 'IMAGERY'
-->
<query>
 <for value="collections"/>
 <dataCenterId>
   <list><value>ORNL_TS1</value></list>
 </dataCenterId>
 <where>
    <collectionCondition>
      <parameter>
        <textPattern>'%Imagery%'</textPattern>
      </parameter>
    </collectionCondition>
 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
   Search for collections from GSFCECS and ORNL that have
   processing level 1A or 2
-->
<query>
 <for value="collections"/>
  <dataCenterId>
      <list>
         <value>GSFCECS</value>
         <value>ORNL_TS1</value>
      </list>
  </dataCenterId>
 <where>
  <collectionCondition negated="y">
    <processingLevel><list>
         <value>'1A'</value>
         <value>'2'</value>
         </list></processingLevel>
  </collectionCondition>
 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
   Search for collections from ORNL with:
   temporal range: periodic range between Jan 1, 1990 and Dec. 31 1998
                from the 1st to the 300th day of each year, AND
   spatial extent: bounding box 60S, 70W to 60N, 70E.
-->
<query>
 <for value="collections"/>
  <dataCenterId>
     <value>ORNL_TS1</value>
  </dataCenterId>
 <where>
  <collectionCondition>
   <temporal>
      <startDate><Date YYYY="1990" MM="01" DD="01"/></startDate>
      <stopDate><Date YYYY="1998" MM="12" DD="31"/></stopDate>
      <startDay value="1"/>
      <endDay value="300"/>
   </temporal>
  </collectionCondition>
  <collectionCondition negated="n">
    <spatial operator="RELATE">
       <IIMSPolygon>
         <IIMSLRing>
           <IIMSPoint  long='-10' lat='85'/>
           <IIMSPoint  long='10' lat='85'/>
           <IIMSPoint  long='10' lat='89'/>
           <IIMSPoint  long='-10' lat='89'/>
           <IIMSPoint  long='-10' lat='85'/>
         </IIMSLRing>
       </IIMSPolygon>    </spatial>
  </collectionCondition>
 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for collections from ORNL with
temporal range: periodic range between Jan 1, 1990 and Dec. 31 1998
                from the 1st to the 300th day of each year, AND
                some days of January.
source name: L7 or AM-1 AND
spatially covering any 'temperate' region or USA
-->
```

```
<query>
 <for value="collections"/>
  <dataCenterId>
     <list><value>ORNL-_TS1/value></list>
  </dataCenterId>
 <where>
  <collectionCondition>
   <temporal>
      <startDate><Date YYYY="1990" MM="01" DD="01"/></startDate>
      <stopDate><Date YYYY="1998" MM="12" DD="31"/></stopDate>
      <startDay value="1"/>
      <endDay value="300"/>
   </temporal>
  </collectionCondition>
  <collectionCondition negated='n'>
     <sourceName>
        <list>
           <value>'L7'</value>
           <value>'AM-1'</value>
        </list>
     </sourceName>
  </collectionCondition>

  <collectionCondition>
    <spatialKeywords>
        <list>
           <value>'temperate'</value>
           <value>'USA'</value>
        </list>
    </spatialKeywords>
  </collectionCondition>

  <collectionCondition>
    <temporalKeywords>
       <textPattern>'%january%'</textPattern>
    </temporalKeywords>
  </collectionCondition>
 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
        <for value="collections"/>
        <dataCenterId>
                <value>ORNL_TS1</value>
        </dataCenterId>
        <where>
           <collectionCondition>
                <psaNames>
               <list>
                     <value>'Provider_Specific_Attribute_1'</value>
                 <value>'Provider_Specific_Attribute_3'</value>
```

```
              </list>
               </psaNames>
          </collectionCondition>
       </where>
</query>
```

Code Example 129.   Queries for Granules (Inventory Search)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules from the L70R or L70RWRS or GLCF_GRANULE_METADATA datasets that have browse data and were categorized
as Day granules.
-->
<query>
 <for value="granules"/>
  <dataCenterId>
     <list><value>ORNL_TS1</value></list>
  </dataCenterId>
 <where>
 <granuleCondition><browseOnly/>
 </granuleCondition>

 <granuleCondition>
    <cloudCover>
       <range lower="10" upper="20"/>
    </cloudCover>
</granuleCondition>

 <granuleCondition><dataSetId>
    <list>
       <value>'L70R'</value>
       <value>'L70RWRS'</value>
       <value>'GLCF_GRANULE_METADATA'</value>
    </list>
 </dataSetId></granuleCondition>

 <granuleCondition><dayNightFlag value="DAY"/>
 </granuleCondition>

 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules the specified spatial region, and periodic temporal extent
-->
<query>
 <for value="granules"/>
  <dataCenterId>
     <list><value>ORNL_TS1</value></list>
  </dataCenterId>
 <where>
  <granuleCondition>
   <spatial operator="RELATE">
     <Polygon>
       <LRing>
         <CList>-120, -30, -100, -60, 5, -90, 5, 85, -120, 30, -120, -30</CList>
       </LRing>
       <LRing>
         <CList>80, 20, 80, 60, 20, 60, 20, 20, 80, 20</CList>
       </LRing>
     </Polygon>
   </spatial>
 </granuleCondition>

 <granuleCondition>
   <temporal>
        <startDate><Date YYYY="1980" MM="01" DD="01"/></startDate>
        <stopDate><Date YYYY="1998" MM="12" DD="31"/></stopDate>
        <startDay value="1"/>
        <endDay value="300"/>
   </temporal>
 </granuleCondition>

 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules from ORNL with source name SWIR or TIR
-->
<query>
 <for value="granules"/>
 <dataCenterId>
   <list><value>ORNL_TS1</value></list>
 </dataCenterId>
 <where>
 <granuleCondition>
   <sensorName><list>
         <value>'SWIR'</value>
         <value>'TIR'</value></list>
   </sensorName>
 </granuleCondition>

 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules with temporal range: periodic range between Jan 1, 1990 and Dec. 31 1998 from the 1st to the 300th day
of each year.
-->
<query>
```

```
 <for value="granules"/>
 <dataCenterId><list><value>ORNL_TS1</value></list>
 </dataCenterId>
 <where>
 <granuleCondition>
   <temporal>
       <startDate><Date YYYY="1990" MM="01" DD="01"/></startDate>
       <stopDate><Date YYYY="1998" MM="12" DD="31"/></stopDate>
       <startDay value="1"/>
       <endDay value="300"/>
   </temporal>
 </granuleCondition>
 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules in the specified list of granule ID s AND in the project/campaign 'rivers' and with PATH=15 and ROW =
33 for WRS1 type.
-->
<query>
 <for value="granules"/>
  <dataCenterId>
     <list><value>ORNL_TS1</value><value>GSFCECS</value></list>
  </dataCenterId>
 <where>
 <granuleCondition>
     <CampaignShortName><list>
        <value>'rivers'</value>
     </list></CampaignShortName>
 </granuleCondition>

 <granuleCondition><ECHOGranuleID>
    <list>
       <value>'G_ORNL_941.1'</value>
       <value>'G_ORNL_937.1'</value>
       <value>'G_ORNL_939.1'</value>
       <value>'G_ORNL_768.1'</value>
       <value>'ECS2:GR:2000059170'</value>
       <value>'ECS2:GR:2000055252'</value>
       <value>'ECS2:GR:2000058193'</value>
       <value>'ECS2:GR:2000059012'</value>
       <value>'ECS2:GR:2000058825'</value>
       <value>'ECS2:GR:2000059048'</value>
    </list>
 </ECHOGranuleID></granuleCondition>

 <granuleCondition>
    <pathRow>
        <path><range lower='15' upper='15'/></path>
        <row><range lower='33' upper='33'/></row>
        <wrsType value='1'/>
    </pathRow>
 </granuleCondition>

 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
        <for value="granules"/>
        <dataCenterId>
                <value>ORNL_TS1</value>
        </dataCenterId>
        <where>
           <granuleCondition>
                <psa>
                   <psaName>'COORDINATE_UNITS_NAME'</psaName>
                   <psaValue><value>'METERS'</value></psaValue>
                </psa>
           </granuleCondition>
        </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
   <for value="granules"/>
   <dataCenterId>
      <value>ORNL_TS1</value>
   </dataCenterId>
   <where>
      <granuleCondition>
         <psa>
            <psaName>'SAMPLE_DATE'</psaName>
               <psaValue>
                  <dateRange>
                     <startDate>
                       <Date YYYY="2000" MM="05" DD="12"/></startDate>
                     <stopDate>
                       <Date YYYY="2000" MM="10" DD="12"/>
                     </stopDate>
                  </dateRange>
               </psaValue>
             </psa>
      </granuleCondition>
    </where>
</query>
```

6.1.6.1 QueryRequest

Below is an example QueryRequest with nothing specified for PresentationDescription (taking default values) and its response with results. Note that the actual result string is embedded in a wrapper XML message within a CDATA field, so that even though it looks like XML, it will have to be extracted in order to parse it as XML, which is done to accommodate non-XML return formats.

Code Example 130. Query Request

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <QueryRequest>
      <QueryExpression>
         <query>
<![CDATA[

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
 <for value="granules"/>
  <dataCenterId>
     <value>ORNL_TS1</value>
  </dataCenterId>
 <where>
  <granuleCondition>
     <dataSetId><textPattern>'%Global%'</textPattern></dataSetId>
  </granuleCondition>
 </where>
</query>

]]>
         </query>
         <namespace>none</namespace>
         <QueryLanguage>
            <IIMSAQL/>
         </QueryLanguage>
      </QueryExpression>
      <ResultType>
         <RESULTS/>
      </ResultType>
            <IteratorSize>2</IteratorSize>
   </QueryRequest>
</CatalogService>
```

Code Example 131. Query Response

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
  <QueryResponse>
    <BooleanResult>
      <BooleanResultType>
        <REQUEST_SUCCEEDED/>
      </BooleanResultType>
    </BooleanResult>
    <ReturnData>
    <MessageFormat>
      <XML/>
    </MessageFormat>
    <payload>
      <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "http://api.echo.nasa.gov/echo/dtd/ECHOGranuleResults.dtd">
<results>
  <provider name='ORNL_TS1'>
    <result number='1' itemId='G2530-ORNL'>
      <GranuleURMetaData>
        <ECHOItemId>G2530-ORNL</ECHOItemId>
        <GranuleUR>DUNNESOIL.soilwhc.dat</GranuleUR>
        <ECHOInsertDate>2002-04-10 14:56:40.0</ECHOInsertDate>
        <ECHOLastUpdate>2002-03-20 14:56:44.0</ECHOLastUpdate>
        <GranuleShortName>soilwhc.dat</GranuleShortName>
        <AccessConstraint>NOT_ACCESS_RESTRICTED</AccessConstraint>
        <Price>0</Price>
        <CatalogItemId>G2530-ORNL</CatalogItemId>
        <CollectionMetaData>
          <ShortName>GLOBAL DISTRIBUTION OF PLANT-EXTRACTABLE WATER CAPACITY OF SOIL (DUNNE)</ShortName>
          <VersionID>0</VersionID>
          <DataSetId>GLOBAL DISTRIBUTION OF PLANT-EXTRACTABLE WATER CAPACITY OF SOIL (DUNNE)</DataSetId>
        </CollectionMetaData>
        <DataGranule>
          <SizeMBDataGranule>73715</SizeMBDataGranule>
          <ProductionDateTime>1980-06-01 00:00:00.0</ProductionDateTime>
        </DataGranule>
        <RangeDateTime>
          <RangeEndingTime>00:00:00</RangeEndingTime>
          <RangeEndingDate>1996-12-31 00:00:00.0</RangeEndingDate>
          <RangeBeginningTime>00:00:00</RangeBeginningTime>
          <RangeBeginningDate>1996-01-01 00:00:00.0</RangeBeginningDate>
        </RangeDateTime>
        <SpatialDomainContainer>
          <HorizontalSpatialDomainContainer>
<BoundingRectangle><WestBoundingCoordinate>-
180</WestBoundingCoordinate><NorthBoundingCoordinate>90</NorthBoundingCoordinate><EastBoundingCoordinate>180</EastBoundingC
oordinate><SouthBoundingCoordinate>-90</SouthBoundingCoordinate></BoundingRectangle>

          </HorizontalSpatialDomainContainer>
        </SpatialDomainContainer>
        <MeasuredParameter>
          <MeasuredParameterContainer>
            <ParameterName>SOIL MOISTURE/WATER CONTENT SOIL WATER HOLDING CAPACITY</ParameterName>
          </MeasuredParameterContainer>
        </MeasuredParameter>
        <StorageMediumClass>
          <StorageMedium>On-Line</StorageMedium>
        </StorageMediumClass>
        <Platform>
          <PlatformShortName>TOPOGRAPHIC MAP</PlatformShortName>
          <Instrument>
            <InstrumentShortName>ANALYSIS</InstrumentShortName>
            <Sensor>
              <SensorShortName>ANALYSIS</SensorShortName>
            </Sensor>
          </Instrument>
        </Platform>
        <Contact>
          <Role>User Services Office</Role>
          <ContactOrganizationName>ORNL DAAC User Services Office</ContactOrganizationName>
          <ContactOrganizationAddress>
            <StreetAddress>Oak Ridge National Laboratory,      P.O. Box 2008, MS 6407</StreetAddress>
            <City>Oak Ridge</City>
            <StateProvince>Tennessee</StateProvince>
            <PostalCode>37831-6407</PostalCode>
            <Country>USA</Country>
          </ContactOrganizationAddress>
          <ContactOrganizationAddress>
            <StreetAddress>Oak Ridge National Laboratory,P.O. Box 2008, MS 6407</StreetAddress>
            <City>Oak Ridge</City>
            <StateProvince>Tennessee</StateProvince>
            <PostalCode>37831-6407</PostalCode>
            <Country>USA</Country>
          </ContactOrganizationAddress>
          <OrganizationTelephone>
            <TelephoneNumber>+1 (865) 241-3952</TelephoneNumber>
            <TelephoneType>Telephone</TelephoneType>
          </OrganizationTelephone>
          <OrganizationEmail>
            <ElectronicMailAddress>ornldaac@ornl.gov</ElectronicMailAddress>
          </OrganizationEmail>
        </Contact>
        <OnlineAccessURLs>
          <OnlineAccessURL>
            <URL>http://daac.ornl.gov/global_soil/Dunne/data/soilwhc.dat</URL>
            <MimeType>text</MimeType>
          </OnlineAccessURL>
        </OnlineAccessURLs>
      </GranuleURMetaData>
    </result>
  </provider>
</results>
]]>
      </payload>
    </ReturnData>
    <RequestID>RGuest7213396921014333628487</RequestID>
    <ResultSetID>RGuest7213396921014333628487</ResultSetID>
    <ResultType>
      <RESULTS/>
    </ResultType>
    <Status>
      <SUCCESS_RESULTS_AVAILABLE/>
    </Status>
    <Hits>1</Hits>
    <Cursor>2</Cursor>
  </QueryResponse>
</CatalogService>
```

6.1.6.2 PresentRequest

Below is a sample PresentRequest with a previously saved result set returning GranuleVersionId, SizeMBData, Price, shortName and sensor information.

Code Example 132. Present request.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <PresentRequest>
   <ResultSetID>result1</ResultSetID>
   <PresentationDescription>
        <TupleType>
         <attributeName>GranuleVersionId</attributeName>
         <PrimitiveTypeName>
          <String/>
         </PrimitiveTypeName>
        </TupleType>
        <TupleType>
         <attributeName>SizeMBDataGranule</attributeName>
         <PrimitiveTypeName>
          <String/>
         </PrimitiveTypeName>
        </TupleType>
        <TupleType>
         <attributeName>Price</attributeName>
         <PrimitiveTypeName>
          <String/>
         </PrimitiveTypeName>
        </TupleType>
        <TupleType>
         <attributeName>shortName</attributeName>
         <PrimitiveTypeName>
          <String/>
         </PrimitiveTypeName>
        </TupleType>
        <TupleType>
         <attributeName>sensor</attributeName>
         <PrimitiveTypeName>
          <String/>
         </PrimitiveTypeName>
        </TupleType>
        <DTDType>
             <ECHO/>
         </DTDType>
       </PresentationDescription>
      <IteratorSize>2</IteratorSize>
     <Cursor>1</Cursor>
  </PresentRequest>
</CatalogService>
```

Code Example 133. Present response.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
  <PresentResponse>
    <BooleanResult>
      <BooleanResultType>
        <REQUEST_SUCCEEDED/>
      </BooleanResultType>
    </BooleanResult>
    <ReturnData>
    <MessageFormat>
      <XML/>
    </MessageFormat>
    <payload>
      <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "-//ECHO GranuleResults (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/ECHOGranuleResults.dtd">
<results>
  <provider name='ORNL_TS1'>
    <result number='1' itemId='G5432-ORNL_TS1'>
      <GranuleURMetaData>
      <ECHOItemId>G5432-ORNL_TS1</ECHOItemId>           <GranuleUR>BOREAS_AES5DAVG.canadian_5_day_avg_daily.dat</GranuleUR>
        <shortName>canadian_5_day_avg_daily.dat</shortName>
        <SizeMBDataGranule>4649911</SizeMBDataGranule>
        <sensor name='HUMAN OBSERVER (1001)'/>
        <sensor name='ANEMOMETER (1010)'/>
        <sensor name='BAROMETER (1018)'/>
        <sensor name='DEWPOINT HYDROMETER (1032)'/>
        <sensor name='DRY BULB THERMOMETER (1037)'/>
        <sensor name='RAIN GAUGE (1105)'/>
        <sensor name='SNOW MEASURING ROD (1111)'/>
        <sensor name='WET BULB THERMOMETER (1158)'/>
      </GranuleURMetaData>
    </result>
    <result number='2' itemId='G5444-ORNL_TS1'>
      <GranuleURMetaData>
      <ECHOItemId>G5444-ORNL_TS1</ECHOItemId>         <granuleUR>BOREAS_AES5DAVG.canadian_5_day_avg_hourly.dat</granuleUR>
        <shortName>canadian_5_day_avg_hourly.dat</shortName>
        <SizeMBDataGranule>27547674</SizeMBDataGranule>
        <sensor name='HUMAN OBSERVER (1001)'/>
        <sensor name='ANEMOMETER (1010)'/>
        <sensor name='BAROMETER (1018)'/>
        <sensor name='DEWPOINT HYDROMETER (1032)'/>
        <sensor name='DRY BULB THERMOMETER (1037)'/>
        <sensor name='RAIN GAUGE (1105)'/>
        <sensor name='SNOW MEASURING ROD (1111)'/>
        <sensor name='WET BULB THERMOMETER (1158)'/>
      </GranuleURMetaData>
    </result>
    <Dictionary name='sensor'>
      <sensor name='HUMAN OBSERVER (1001)'>
```

```
        <sensorShortName>HUMAN OBSERVER</sensorShortName>
        <sensorLongName>HUMAN OBSERVER</sensorLongName>
      </sensor>
      <sensor name='ANEMOMETER (1010)'>
        <sensorShortName>ANEMOMETER</sensorShortName>
        <sensorLongName>ANEMOMETER</sensorLongName>
      </sensor>
      <sensor name='BAROMETER (1018)'>
        <sensorShortName>BAROMETER</sensorShortName>
        <sensorLongName>BAROMETER</sensorLongName>
      </sensor>
      <sensor name='DEWPOINT HYDROMETER (1032)'>
        <sensorShortName>DEWPOINT HYDROMETER</sensorShortName>
        <sensorLongName>DEWPOINT HYDROMETER</sensorLongName>
      </sensor>
      <sensor name='DRY BULB THERMOMETER (1037)'>
        <sensorShortName>DRY BULB THERMOMETER</sensorShortName>
        <sensorLongName>DRY BULB THERMOMETER</sensorLongName>
      </sensor>
      <sensor name='RAIN GAUGE (1105)'>
        <sensorShortName>RAIN GAUGE</sensorShortName>
        <sensorLongName>RAIN GAUGE</sensorLongName>
      </sensor>
      <sensor name='SNOW MEASURING ROD (1111)'>
        <sensorShortName>SNOW MEASURING ROD</sensorShortName>
        <sensorLongName>SNOW MEASURING ROD</sensorLongName>
      </sensor>
      <sensor name='WET BULB THERMOMETER (1158)'>
        <sensorShortName>WET BULB THERMOMETER</sensorShortName>
        <sensorLongName>WET BULB THERMOMETER</sensorLongName>
      </sensor>
    </Dictionary>
  </provider>
</results>]]>
    </payload>
    </ReturnData>
    <Cursor>3</Cursor>
    <Hits>821</Hits>
    <Status>
      <SUCCESS_RESULTS_AVAILABLE/>
    </Status>
  </PresentResponse>
</CatalogService>
```

6.2 Provider Profile Service

The provider profile service is for users to retrieve a providers' profile, such as the provider's ID in ECHO, the organization names of providers, contact information of providers, and transactions supported by providers. The provider’s ID is particularly useful when creating a query. Any user could use this service to get the provider's information.

6.2.1 Transactions

6.2.1.1 ListAllProviders

This transaction lists all provider IDs in ECHO.

Code Example 134. ListAllProviders transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderProfileService PUBLIC "-//ECHO ProviderProfileService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderProfileService.dtd">
<ProviderProfileService>
   <ListAllProvidersRequest/>
</ProviderProfileService>
```

6.2.1.2 PresentProviderProfiles

This transaction presents provider's profile, including organization name, spatial projection type, description of holdings provider has, all of provider's contact information.

Code Example 135. PresentProvderProfile transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderProfileService PUBLIC "-//ECHO ProviderProfileService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderProfileService.dtd">
<ProviderProfileService>
   <PresentProviderProfilesRequest>
      <ProviderID>ORNL_TS1</ProviderID>
   </PresentProviderProfilesRequest>
</ProviderProfileService>
```

6.2.1.3 PresentProviderSupportedTransactions

This transaction presents the transactions that the provider supports. Currently there are just some order-related transactions (e.g., order submit, order cancel).

Code Example 136. PresentProvderSupportedTransactions transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderProfileService PUBLIC "-//ECHO ProviderProfileService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderProfileService.dtd">
<ProviderProfileService>
   <PresentProviderSupportedTransactionsRequest>
      <ProviderID>ORNL_TS1</ProviderID>
   </PresentProviderSupportedTransactionsRequest>
</ProviderProfileService>
```

6.3 Order Entry Service

The first step in the order process is creating the order. Though it can look like a very simple transaction, a user needs to understand the parts that make up an order.

An order is a collection of order line items. Each order line item consists of a catalog item ID, the quantity of that catalog item, and all the options associated with that catalog item. A single order may consist of many catalog items that belong to many different providers. Each provider may have its own validation scheme and rules for creating, validating and submitting their particular catalog item. To ensure that appropriate validation happens – as well as to demarcate the larger order for sending to each provider – orders can also be split up into smaller provider orders, sometimes called sub-orders. Each provider order is nothing more than all the order line items in a particular order that belong to a specific provider. A provider order is uniquely identified by a provider order ID. A provider order

ID is the aggregation of the actual provider ID of the provider in question, and the order ID of the order that contains the catalog items for this provider.

The most common way to obtain the catalog items needed for creating an order is through the query transactions found in Catalog Service. ECHO uses its own query language called AQL to search for different catalog items. To learn more about the Catalog Service and the actual Query transaction, please refer to Catalog Service section. To obtain the catalog items, define a query in AQL and pass that query to a transaction like the Query transaction in the Catalog Service (using the QueryExpression parameter). You can also specify in these query transactions exactly how the results should be displayed as well. In the displaying query results, make sure that the 'CatalogItemID' tag is displayed. The string value under this tag is the actual catalog item ID that can be used to order things from the system.

Once the list of catalog items for ordering is obtained, one needs to also know the available or required options for each catalog item that is to be ordered. To list the possible options both allowed and required for each catalog item, use the PresentCatalogItem transaction in the OrderEntryService. After you identify the catalog item ID in the transaction, ECHO will return the complete list of possible options (both required and optional) that one can select for this catalog item.

Note: Sometimes a catalog item may not be orderable (for more information, please refer to the section Restriction on order items and Deleted items covered later in the OrderEntryService). If this is the case, only the catalog item ID and an ErrorMessage is displayed for this item.

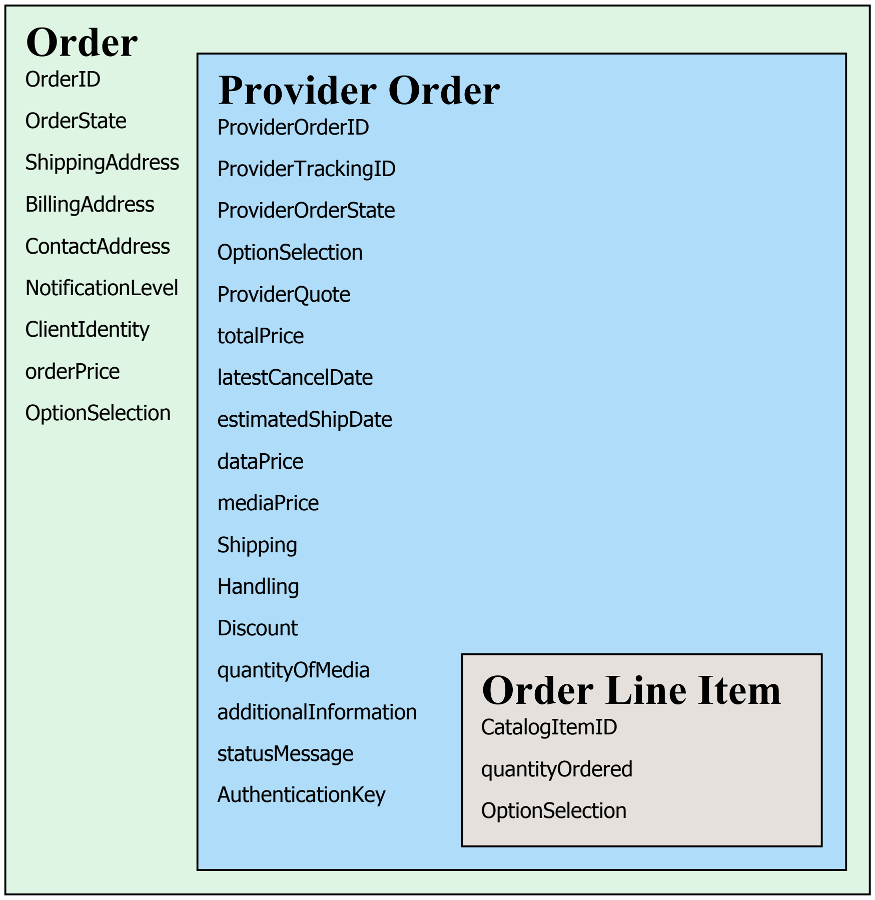


Figure 8. Contents of an Order

Code Example 137. PresentCatalogItem transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <PresentCatalogItemRequest>
      <CatalogItemID>G2076-ORNL</CatalogItemID>
   </PresentCatalogItemRequest>
</OrderEntryService>
```

The option hierarchy in ECHO was created to be as flexible as possible in terms of being able to deal with any kind of option. Because of this flexibility, it can be challenging to create the right option selection hierarchy for each catalog item. Initially, one needs to run the PresentCatalogItem transaction, and from that obtain each possible OptionDefinition that it produces. From the OptionDefinition, one needs to make sure that the option selection created for that catalog item follows the option hierarchy that was defined in that transaction. It is necessary to keep track of all the possible sub-options and sub-sub-options possible for each option. To actually create the option selections for a catalog item, take all the fields in the necessary option definitions (specifically the OptionName element) and map them into the similar hierarchy found in the OptionSelection element, filling in the appropriate Value elements when necessary. All of the information necessary to create an order is complete once the OptionSelection elements for each catalog item in question is filled.

Note: It is not always required to attach options to a certain catalog item. However, if there are any options that are required by that catalog item, they will have to be filled out before one validates and/or submits the order (or else an error will be returned). To learn more about options and how to use them throughout ECHO, refer to the appendices of this document.

One of ECHO's main tasks is to act as a full functioning order creation and tracking server. In designing the system, flexibility and completeness were top priorities. In keeping these goals in mind, the following aspects of the ordering service were enabled:

- Support for ordering catalog items from a variety of different providers.
- Ability to modify and change a variety of options that may exist for an order, provider-order, and/or catalog item.

Note: An order may be modified as soon as it is created, but modifications are not permitted after the order has been submitted.

- Allowing asynchronous flexibility in how and when the user creates a valid and complete order (i.e., the user is not mandated to fill out a complete order in one sitting)

There is some complexity involved in fully creating and submitting an order. The system allows flexibility in how an order is made. A general workflow for creating, validating, and submitting an order is useful in simplifying the process. The figure below illustrates the most commonly used and direct approach to creating and submitting an order.

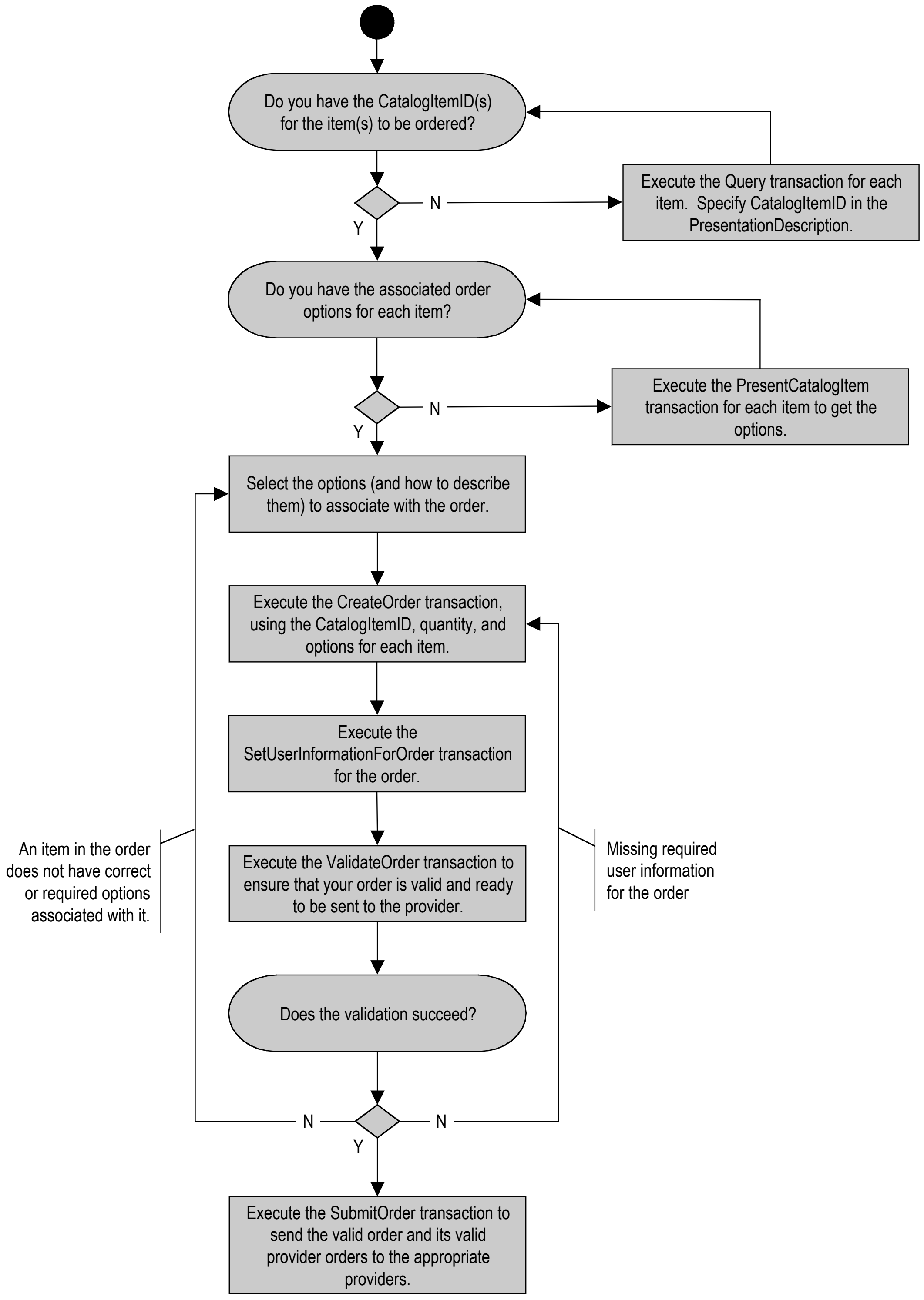

Do you have the CatalogItemID(s) for the item(s) to be ordered?
N
Y
Execute the Query transaction for each item. Specify CatalogItemID in the PresentationDescription.
Do you have the associated order options for each item?
N
Y
Execute the PresentCatalogItem transaction for each item to get the options.
Select the options (and how to describe them) to associate with the order.
Execute the CreateOrder transaction, using the CatalogItemID, quantity, and options for each item.
Execute the SetUserInformationForOrder transaction for the order.
An item in the order does not have correct or required options associated with it.
Execute the ValidateOrder transaction to ensure that your order is valid and ready to be sent to the provider.
Missing required user information for the order
Does the validation succeed?
N
N
Y
Execute the SubmitOrder transaction to send the valid order and its valid provider orders to the appropriate providers.

Figure 9. Creating and Submitting an Order

6.3.1 Establishing a valid client connection

All clients interfacing with the system are required to pass client identifier information to ECHO. ECHO will then take this identifier and attach it to any order submitted through the respective client.

NOTE: If client identifier information is not provided, submitted orders will fail.

The client identifier is a short description and/or name of the client. Client developers are encouraged to keep this information as concise as possible, and to work with the ECHO Operations Group to create an appropriate identifier.

The following Java code example can be used to to create and pass client identifier information. The client should execute this code immediately after an ECHO session has been established. After creating an instance of echoToken, the client can then connect to the desired system (soap URL), such as api.echo.nasa.gov or other system, and call the identify method to pass the client identifier information. This issue must be addressed by the client developer and is essential to the proper submittal of a user order.

```
_echoToken = new EchoToken();
_echoToken.connectTo(soapURL);

try
{
  _echoToken.identify("Client identifier");
}
catch (XMLServiceException e)
{
  Logger.logException(e, SoapServer.class, Logger.ERROR);
}
```

6.3.2 Introduction to Options

The ECHO application frameworks include a mechanism that allows properties to be dynamically defined and attached to entities within the system. A property is a named value, similar to an attribute. The "properties" mechanism has been created to enable easy, code-free extensibility of ECHO entities. Order and User Profile objects currently rely upon this properties mechanism to enable portions of their state to be dynamically defined and set.

The properties mechanism is useful for managing properties that are defined and used by systems that are outside of the context of ECHO. A data provider that ECHO interacts with is a good example of such an external system. The data provider may require that specific information accompany any order or request for data. Furthermore, portions of that information may be uniquely specific to requests for data sent to that particular provider. This is a situation where applying the property mechanism makes sense. Here, the provider-specific information will be defined and stored abstractly as a collection of properties that must be set by the user during the order entry process.

The ECHO API uses the parameters Option and OptionSelection to communicate property definitions and property values respectively. An Option parameter defines a property value that may be set through the ECHO API. An OptionSelection parameter is used to set the value that is defined by the option or to communicate the value that is currently set for the option. The diagram below shows the structure and relationships of the option parameters, as defined in the XML API.

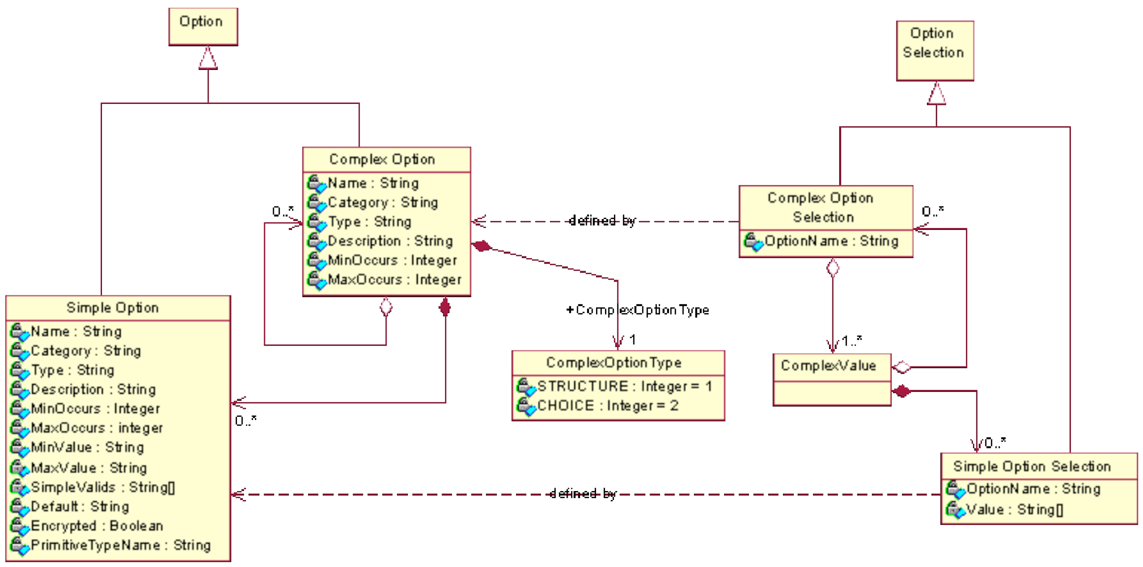


Figure 10. Structure and relationships of the Option Parameters.

There are two types of options: simple and complex. A simple option describes an atomic piece of information of a primitive type. The "Type" field indicates the primitive value type that is expected. A simple option may also specify that multiple instances of that type may be input--similar to an array. The "MinOccurs" and "MaxOccurs" field values define the multiplicity of a corresponding value, while the MinValue and MaxValue fields define the range of allowable values. A simple option may declare the complete set of distinct valid values that may be set for a simple option selection of that type. These are defined in the "Valids" field. Finally, a default value may be specified in the "Default" field.

The second type of option is ComplexOption. A complex option is an option that defines a grouping of input parameters, sub-options, and/or choices. Like simple options, complex options may declare the allowable multiplicity of values. These are defined in the "MinOccurs" and "MaxOccurs" fields. The complex option "Type" field indicates the validation behavior that will be applied to the corresponding complex value instance. There are two types of complex values: structure and choice. A "choice" type indicates that for each complex option selection value (remember that there may be many values), a sub-value for only one of the sub-options contained by the complex option may be set. A "structure" type, on the other hand, allows a complex value to be set that contains set sub-values for each sub-option defined by the complex option.

An OptionSelection is a structure that represents the values that correspond to an Option. Mirroring the option hierarchy, there are two types of option selection: SimpleOptionSelection, and ComplexOptionSelection. A simple option selection has a name that matches the name of its simple option, and contains one or more primitive values. A complex option selection also has a name that matches the name of its [complex] option, and contains one or more complex values. A complex value is a container of option selections. It will contain zero or more simple options and complex options, as defined by the complex option and its associated "Type."

Due to the reflexive nature of options and option selections, reading and generating appropriate OptionSelections can be confusing. An example definition with some explanations and mapping between the definition and the selection has been provided in Appendix B.

6.3.3 Transactions

6.3.3.1 CreateOrder

This transaction initiates the order process. Below is an example of an order without options:

Code Example 138. Order without options.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <CreateOrderRequest>
      <OrderLineItem>
         <CatalogItemID>
            2000054170
         </CatalogItemID>
         <quantityOrdered>
            2
         </quantityOrdered>
      </OrderLineItem>
   </CreateOrderRequest>
</OrderEntryService>
```

To create an order with associated options, here is an annotated example:

Code Example 139. Order with options.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <CreateOrderRequest>
      <OrderLineItem>
         <CatalogItemID>
         5054
      </CatalogItemID>
         <quantityOrdered>
         2
      </quantityOrdered>
<!--
Option Selection For Product 5054
Product 5054 can be ordered using one of three production options (Ancillary, Native, and Production History). Also it can
be received via CD-ROM or FTP Push.
-->
         <OptionSelection>
            <ComplexOptionSelection>
<!--
Root option selection node; the Option Selection must fulfill the constraints defined within the corresponding Option (see
Appendix A).
-->
               <OptionName>ECS-TEST PKG1</OptionName>
<!--
Production Option of 'Native Granule' selected
-->
               <ComplexValue>
                  <SimpleOptionSelection>
                     <OptionName>Production Option</OptionName>
                     <Value>Native Granule</Value>
                  </SimpleOptionSelection>
```

```
<!--
Media Type choice root
-->
                  <ComplexOptionSelection>
                     <OptionName>Media Type</OptionName>
                     <ComplexValue>
<!--
CDROM media type selected
-->
                        <ComplexOptionSelection>
                           <OptionName>CDROM</OptionName>
                           <ComplexValue>
<!--
Compression type of 'Gzip' selected
-->
                              <SimpleOptionSelection>
                                 <OptionName>Compression</OptionName>
                                 <Value>GZip</Value>
                              </SimpleOptionSelection>
<!--
DDISTMEDIAFMT value of 'Rockridge' selected
-->
                              <SimpleOptionSelection>
                                 <OptionName>DDISTMEDIAFMT</OptionName>
                                 <Value>Rockridge</Value>
                              </SimpleOptionSelection>
                           </ComplexValue>
                        </ComplexOptionSelection>
                     </ComplexValue>
                  </ComplexOptionSelection>
               </ComplexValue>
            </ComplexOptionSelection>
         </OptionSelection>
      </OrderLineItem>
   </CreateOrderRequest>
</OrderEntryService>
```

6.3.3.2 AddOrderLineItem

This transaction is used to add one or more order line items to an order. Since each order line item uniquely represents a catalog item with the intention to be ordered, one cannot use this transaction to update an already existing order line item. To modify an order line item within an order, such as changing the quantity and options, use the transaction UpdateOrderLineItem.

Code Example 140. Adding two order line items without options.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <AddOrderLineItemRequest>
      <OrderID>3103</OrderID>
      <OrderLineItem>
         <CatalogItemID>758</CatalogItemID>
         <quantityOrdered>3</quantityOrdered>
      </OrderLineItem>
      <OrderLineItem>
         <CatalogItemID>759</CatalogItemID>
         <quantityOrdered>4</quantityOrdered>
      </OrderLineItem>
   </AddOrderLineItemRequest>
</OrderEntryService>
```

Code Example 141. Adding a single order line item with options.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <AddOrderLineItemRequest>
      <OrderID>1194182309</OrderID>
      <OrderLineItem>
         <CatalogItemID>G2079-ORNL</CatalogItemID>
         <quantityOrdered>4</quantityOrdered>
         <OptionSelection>
            <ComplexOptionSelection>
               <OptionName>ORNL Package Options</OptionName>
               <ComplexValue>
                  <ComplexOptionSelection>
                     <OptionName>Media Type</OptionName>
                     <ComplexValue>
                        <ComplexOptionSelection>
                           <OptionName>FTP_Pull</OptionName>
                           <ComplexValue>
                              <SimpleOptionSelection>
                                <OptionName>Compression</OptionName>
                                <Value>TAR</Value>
                              </SimpleOptionSelection>
                           </ComplexValue>
                        </ComplexOptionSelection>
                     </ComplexValue>
                  </ComplexOptionSelection>
               </ComplexValue>
            </ComplexOptionSelection>
         </OptionSelection>
      </OrderLineItem>
   </AddOrderLineItemRequest>
</OrderEntryService>
```

6.3.3.3 DeleteOrder

This transaction is used to delete one or more orders from the system. A user can only delete orders that are in the pre-submittal phase, i.e., orders that have not yet been submitted. For post-submittal orders, a registered user can use the CancelOrder operation in the User Account Service to cancel an open order. Guest orders cannot be deleted in the post-submittal phase. This transaction will completely delete the entire order, including all associated provider orders. To delete a specific provider order(s), use the DeleteProviderOrder transaction.

Code Example 142. Example of deleting three orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <DeleteOrderRequest>
      <OrderID>1234</OrderID>
      <OrderID>2345</OrderID>
      <OrderID>3456</OrderID>
   </DeleteOrderRequest>
</OrderEntryService>
```

6.3.3.4 DeleteOrderLineItem

This transaction is used to remove one or more order line items from an order.

Code Example 143. Deleting two order line items from an order.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <DeleteOrderLineItemRequest>
      <OrderID>3103</OrderID>
      <CatalogItemID>758</CatalogItemID>
      <CatalogItemID>759</CatalogItemID>
   </DeleteOrderLineItemRequest>
</OrderEntryService>
```

6.3.3.5 DeleteProviderOrder

This transaction is used to remove a provider order from a larger order context. It will only delete the specified provider order (and thus all of the order line items associated with that provider) from the order. All other non-specified provider orders will stay intact.

Code Example 144. Deleting a specific provider order from an order.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <DeleteProviderOrderRequest>
      <ProviderOrderID>
         <ProviderID>ECHO-TEST</ProviderID>
         <OrderID>3955</OrderID>
      </ProviderOrderID>
   </DeleteProviderOrderRequest>
</OrderEntryService>
```

6.3.3.6 ListUnsubmittedOrderSummary

This transaction is used to present the user with a summary of orders that have been created but not yet submitted, and matches one of the specified order states passed to it. This transaction only returns the ID and state of each order that meets the criteria. If no unsubmitted order state is passed to the transaction, then all the orders that have been created but not yet submitted for that user will be presented.

> *Note: This is one of the few transactions in the OrderEntryService that a guest does not have access to. Only a registered user (this does not include providers) can use this transaction.*

Code Example 145.   Presenting orders in two unsubmitted states: NOT_VALIDATED and VALIDATED.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <ListUnsubmittedOrderSummaryRequest>
      <UnsubmittedOrderState>
         <NOT_VALIDATED/>
      </UnsubmittedOrderState>
      <UnsubmittedOrderState>
         <VALIDATED/>
      </UnsubmittedOrderState>
   </ListUnsubmittedOrderSummaryRequest>
</OrderEntryService>
```

Code Example 146.   Presenting orders in all unsubmitted states.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <ListUnsubmittedOrderSummaryRequest/>
</OrderEntryService>
```

6.3.3.7 PresentCatalogItem

This transaction is used to obtain information about one or more catalog items. The information available for a catalog item includes

- Type and description of the item,
- Which provider owns it,
- Price (if any),
- Options (required and available) for the item. These options may include shipping and packaging options, or the information necessary for the fulfillment of a service.

Each ordered catalog item creates one order line in a user's order.

Code Example 147.   Presenting a catalog item.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <PresentCatalogItemRequest>
      <CatalogItemID>G2076-ORNL</CatalogItemID>
   </PresentCatalogItemRequest>
</OrderEntryService>
```

6.3.3.8 PresentOptionDefinitionsForOrder

This transaction is used to request a list of the options (required and available) that may be selected for a particular order. These options may include shipping and packaging options, and/or payment information. They apply to all the provider orders within the order.

Note: This transaction is not fully implemented for ECHO release 6.0.

6.3.3.9 PresentOptionDefinitionsForProviderOrder

This transaction is used to present the provider-level option selections for a specific provider order. These selections may include shipping and packaging options, and will be applied to all order line items that contain provider-specific options within that provider order. If no option names are declared, all the options are displayed. Use the PresentOrderOptionDefinitionsForProvider transaction to view the possible options one can set for this provider. Use the SetProviderOptionSelectionsForProviderOrder transaction to set the options.

Code Example 148. Presenting option selections for provider order.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <PresentProviderOptionSelectionsForProviderOrderRequest>
      <ProviderOrderID>
         <ProviderID>P14381</ProviderID>
         <OrderID>1731053521</OrderID>
      </ProviderOrderID>
      <OptionName>Special Instructions</OptionName>
   </PresentProviderOptionSelectionsForProviderOrderRequest>
</OrderEntryService>
```

6.3.3.10 PresentOrder

This transaction is used to present the complete specification and description of an order that is being built or to see the status of an order that has already been submitted. The user can use the PresentProviderOrder transaction to see just the portion of the order that is specific to a particular provider.

Code Example 149. Presenting three orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <PresentOrderRequest>
      <OrderID>1234</OrderID>
      <OrderID>2345</OrderID>
      <OrderID>3456</OrderID>
   </PresentOrderRequest>
</OrderEntryService>
```

6.3.3.11 PresentOrderOptionDefinitionsForProvider

This transaction is used to request a list of the options (required and available) that may be selected for a particular provider. These options may include shipping and packaging options, and/or payment information. Currently this transaction is not fully implemented.

6.3.3.12 PresentProviderOrder

This transaction is used to present the portion(s) of an order that is fulfilled by one particular provider. All orders are broken into these sub-orders with each sub-order grouped by a provider.

Code Example 150. Presenting two orders from the same provider.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <PresentProviderOrderRequest>
      <ProviderOrderID>
         <ProviderID>ECHO-TEST</ProviderID>
         <OrderID>3202</OrderID>
      </ProviderOrderID>
      <ProviderOrderID>
         <ProviderID>ECHO-TEST</ProviderID>
         <OrderID>3252</OrderID>
      </ProviderOrderID>
   </PresentProviderOrderRequest>
</OrderEntryService>
```

6.3.3.13 QuoteOrder

This transaction is a request for a particular order to be quoted. A quoted order provides a binding price for the order as a whole. A quote request is a non-blocking request, as the binding quote response from each of the necessary providers may take a day or more to receive. No changes may be made to an order while it is in the quoting process. Also, a change to the order subsequent to the receipt of a quote may invalidate the quoted price.

Code Example 151. Quoting on a single order.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <QuoteOrderRequest>
      <OrderID>2959</OrderID>
   </QuoteOrderRequest>
</OrderEntryService>
```

Notes:

The QuoteOrder transaction only works on validated orders. An order must first be validated through the ValidateOrder transaction before the QuoteOrder transaction can take place.

6.3.3.14 SetAuthenticationKey

This transaction is used to set the name of the authenticator to be used when ordering data from a specific provider.

When a registered user wishes to place an order for data to a provider requiring an authenticator, the user must first create the authenticator that the specified provider recognizes (see Section 2.2.21). The SetAuthenticationKey transaction is then called with the name of the needed authenticator. If a provider requires an authenticator, the SetAuthenticationKey transaction should be called before the order is validated.

Code Example 152. Setting the authentication key.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService SYSTEM "http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
        <SetAuthenticationKeyRequest>
                <ProviderOrderID>
                        <ProviderID>LPDAAC-DAAC</ProviderID>
                        <OrderID>123456789</OrderID>
                </ProviderOrderID>
                <AuthenticatorName>MyLPDAAC_auth</AuthenticatorName>
        </SetAuthenticationKeyRequest>
</OrderEntryService>
```

6.3.3.15 SetOptionSelectionsForOrder

This transaction is used to set a list of options that have been defined for a particular order. These options may include shipping and packaging options, and/or payment information and is applied to all the provider orders within that order. Use PresentOptionDefinitionsForOrder transaction to view the possible options one can set for this order.

Note: This transaction is not fully implemented for ECHO release 6.0.

6.3.3.16 SetOptionSelectionsForProviderOrder

This transaction is used to set the provider-level options for a specific provider order. These options may include shipping and packaging options, and will be applied all order line items that contain provider-specific options within that provider order. Use the PresentOrderOptionDefinitionsForProvider transaction to view the possible options one can set for this provider.

Code Example 153. Set provider option selections for a provider order.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <SetProviderOptionSelectionsForProviderOrderRequest>
      <ProviderOrderID>
         <ProviderID>1000</ProviderID>
         <OrderID>3955</OrderID>
      </ProviderOrderID>
      <OptionSelection>
         <ComplexOptionSelection>
            <OptionName>ECS-TEST PKG1</OptionName>
            <ComplexValue>
               <SimpleOptionSelection>
                  <OptionName>Production Option</OptionName>
                  <Value>Native Granule</Value>
               </SimpleOptionSelection>
               <ComplexOptionSelection>
                  <OptionName>Media Type</OptionName>
                  <ComplexValue>
                     <ComplexOptionSelection>
                        <OptionName>CDROM</OptionName>
                        <ComplexValue>
                           <SimpleOptionSelection>
                              <OptionName>Compression</OptionName>
                              <Value>GZip</Value>
                           </SimpleOptionSelection>
                           <SimpleOptionSelection>
                              <OptionName>DDISTMEDIAFMT</OptionName>
                              <Value>Rockridge</Value>
                           </SimpleOptionSelection>
                        </ComplexValue>
                     </ComplexOptionSelection>
                  </ComplexValue>
               </ComplexOptionSelection>
            </ComplexValue>
         </ComplexOptionSelection>
      </OptionSelection>
   </SetProviderOptionSelectionsForProviderOrderRequest>
</OrderEntryService>
```

6.3.3.17 SetUserInformationForOrder

Every order must have its own user-specific information attached to it so that a data provider can process the order. Contact Address (which includes a user's name, address, phone number, and email address) is required. Billing and shipping address are both optional. Each order is independent of other orders, so user information must be set for every order created.

Again, the system stresses flexibility. User information is not required until you either validate the order or actually submit the order. At any time before submission, the user has the option of providing some or none of the user information needed for the order. The user can also update and delete order line items from a specific order either before or after setting the user information. However, eventually all of this information will be necessary before the provider processes an order. If the required information is not present at the time of order validation or submission, an error WILL be sent to the user. Thus before submitting or validating an order, make sure that the required user information for that particular order is completely filled out. To do this, one must use the SetUserInformationForOrder transaction within the OrderEntryService.

Code Example 154. Set User Information for order

```
<?xml version="1.0"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
 <SetUserInformationForOrderRequest>
  <OrderID>3955</OrderID>
  <ShippingAddress>
   <shipToBusinessName>CompanyName</shipToBusinessName>
   <ContactName>
    <FirstName>Michael</FirstName>
    <LastName>Hudson</LastName>
   </ContactName>
   <specialInstruction>Please leave the package at the front door, not in the mailbox.</specialInstruction>
   <EmailString>mikeh@company.com</EmailString>
   <AddressInformation>
    <AddressID>Shipping</AddressID>
     <USFormat>TRUE</USFormat>
    <Street1>7600 Kingsbury Road</Street1>
    <Street2/>
    <City>Alexandria</City>
```

```
            <State>VA</State>
            <Zip>22315</Zip>
            <Country>USA</Country>
         </AddressInformation>
      </ShippingAddress>
      <BillingAddress>
         <ContactName>
        <FirstName>Michael</FirstName>
        <LastName>Hudson</LastName>
         </ContactName>
         <EmailString>billg@company.com</EmailString>
         <AddressInformation>
            <AddressID>Billing</AddressID>
            <USFormat>TRUE</USFormat>
            <Street1>7600 Kingsbury Road</Street1>
            <Street2/>
            <City>Alexandria</City>
            <State>VA</State>
            <Zip>22315</Zip>
            <Country>USA</Country>
         </AddressInformation>
      </BillingAddress>
      <ContactAddress>
         <ContactName>
        <FirstName>Michael</FirstName>
        <LastName>Hudson</LastName>
         </ContactName>
         <PhoneString>703-921-9393</PhoneString>
         <EmailString>mhudson@company.com</EmailString>
         <AddressInformation>
            <AddressID>Contact</AddressID>
            <USFormat>TRUE</USFormat>
            <Street1>7600 Kingsbury Road</Street1>
            <City>Alexandria</City>
            <State>VA</State>
            <Zip>22315</Zip>
            <Country>USA</Country>
         </AddressInformation>
      </ContactAddress>
   </SetUserInformationForOrderRequest>
</OrderEntryService>
```

6.3.3.18 SubmitOrder

At this point, the order should be complete and valid, with all appropriate options filled out and the associated user information completed. The order may also have been quoted at this point as well. If all the previous actions have executed successfully, and no other changes have been made to the order, then the order is ready to be submitted to the system. To do this, use the SubmitOrder transaction in the OrderEntryService.

Code Example 155. SubmitOrder transaction

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
```

```
<OrderEntryService>
   <SubmitOrderRequest>
      <OrderID>1002</OrderID>
      <NotificationLevel>
         <VERBOSE/>
      </NotificationLevel>
   </SubmitOrderRequest>
</OrderEntryService>
```

The NotificationLevel is the place where a user specifies a preference for receiving the updated status of a particular order. The choices are

- **VERBOSE:** the system will email all provider order state changes and status message updates to the user;
- **DETAIL:** the system will email the user when only provider order state changes are made;
- **INFO:** the system will email the user when the provider order is closed or cancelled;
- **CRITICAL:** the system will email the user when the order fails when submitting or is rejected by the provider;
- **NONE:** the system will not send the user email. The user will have to come back to ECHO and use transactions like 'PresentOrder' to find out about the status of the order.

For any choice, the user can always come back to ECHO and check the status of the order through the 'PresentOrder' transaction. If the user does not specify any value of this data type, for a registered user's order, the user's preference value will be used; for guests' orders, or if no preference value is set for a registered user, the default value is VERBOSE.

Once the order is transmitted to the system using the Submit transaction, each provider order within the greater order is sent to their appropriate providers. It is then up to each provider whether they will accept the provider order and/or how they process the provider order. From this point on, the entire order is out of the hands of the ECHO. However, one can track the status of the order using the PresentOrder transaction in the OrderEntryService. At any point before the order is actually shipped, the user may request cancellation of their order through the CancelOrder transaction in the UserAccountService. The user can then use the PresentOrder transaction again to track whether the order was successfully cancelled.

When ECHO sends a request to a provider to either quote, submit, or cancel a particular order, it is necessary that both ECHO and that provider have the same ID for that order. Within ECHO, the specific provider order is denoted by the ProviderOrderID (which is just the combination of the provider name and the order ID). However, the provider may also have its own ID system for tracking a particular provider order. As a solution, the provider order in ECHO also contains a provider-tracking ID. When the request is sent to the provider from ECHO, the provider has the option to either accept the order ID that ECHO has already given that provider order, or provide ECHO with its own unique ID. If the provider does pass back its own unique ID as the provider-tracking ID, then ECHO will also use that ID in addition to the normal ECHO ProviderOrderID to refer to this provider order. If the provider does not specify a tracking ID, then it is assumed that the order ID that ECHO currently uses will suffice, and the provider order will be referenced using just the ECHO ProviderOrderID instead.

6.3.3.19 UpdateOrderLineItem

This transaction is used to modify one or more order line items, including the quantity and the options associated and available for that order line item.

Code Example 156. Updating quantity of an order line item.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
```

```
   <UpdateOrderLineItemRequest>
      <OrderID>3103</OrderID>
      <OrderLineItem>
         <CatalogItemID>731</CatalogItemID>
         <quantityOrdered>4</quantityOrdered>
      </OrderLineItem>
   </UpdateOrderLineItemRequest>
</OrderEntryService>
```

6.3.3.20 ValidateOrder

One characteristic of the ordering system with ECHO is flexibility. One can create, update, and delete orders at any time as well as add, change, and delete order options at any time. There are no specific restrictions on any of these actions at any time until one decides that the order creation is over and they want to submit the order to the provider. At this time, an order must first be validated. This is to ensure that the order is complete and correct – that all necessary and required options are filled out, and the required user-associated information for that order is appropriately completed. If one attempts to try to submit a non-validated order, an error message will be returned to the user saying as much. Currently, only the first validation error encountered is returned when trying to validate, so revalidation is necessary once the reported error is fixed to fully validate the order. Also, if any aspect of a validated order is changed before it is submitted, then the order becomes non-validated again. To validate an order, use the ValidateOrder transaction in the OrderEntryService (see code example below).

After validation, it may be possible (depending on whether the provider supports this transaction) to get the exact price of each provider order before you actually submit the order. To do this, run the QuoteOrder transaction in the OrderEntryService.

> *Notes:*
>
> *The QuoteOrder transaction only works on validated orders. An order must first be validated through the ValidateOrder transaction before the QuoteOrder transaction can take place.*

Keep in mind that because many of the validation rules can be provider specific and change over time, running the ValidateOrder transaction may often return a validation error. However, the error messages should be complete enough to help you determine what aspects of the order need to be filled out or corrected.

It is important to understand that you may have to run the order through this transaction often, each time obtaining an error message and correcting the problem appropriately. In general, the error messages cover whether required options for a catalog item, provider-order, or order are not filled out, or whether a piece of user information was not completed. Also, error messages can occur if the order specified does not exist, doesn't belong to the current user, or the order itself has already been submitted or quoted.

Code Example 157. ValidateOrder transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <ValidateOrderRequest>
      <OrderID>4104</OrderID>
   </ValidateOrderRequest>
</OrderEntryService>
```

6.3.4 Notifications

ECHO users do not interact with the providers directly. However, actions by users (e.g. submitting, quoting, and canceling an order) require that the providers either accept or reject the action. If such actions are initiated, ECHO will communicate with the providers for the decision. Note that users normally receive a REQUEST SUCCEEDED BooleanResultType in the response message. This only means that ECHO successfully received the request from the user. The acceptance or rejection of the request is the responsibility of the provider. As the response from each associated provider may take a day or more to receive, the status of each provider order and the order at large is not known automatically. Users can track the providers' decisions by looking at the timestamped StatusMessage in the PresentOrder response.

6.3.4.1 QuoteOrder

This transaction is used to request for an order to be quoted. Do not change an order while it is in the quoting process. A change to the order subsequent to the receipt of a quote may invalidate the quoted price. If the quote request is accepted by a specific provider, the supplied quote information will be shown in the ProviderQuote under the ProviderOrder belonging to that provider in the PresentOrder response.

6.3.4.2 SubmitOrder

This transaction is used to place an order. The user can request to be notified of status changes by email. However, to see the order information such as price, expected shipping date, etc., use the PresentOrder transaction.

6.3.4.3 CancelOrder

This transaction, which is in the UserAccountService, is used to cancel an order or a provider order that is processing. There may be a cost associated with canceling the order. Again, users can check the status of the request using the PresentOrder transaction. If a user has requested to be notified of status changes by email at the level VERBOSE, DETAIL, or INFO, the user will be notified once the order is cancelled.

6.3.5 Order Entry Service Error Messages

Transaction	Error Message	Description
AddOrderLineItem	[<OptionName>] is not a valid property for this item	An invalid option is associated with an item.
	gov.nasa.echo.business.ejb.order. OrderException (Cannot add an item with quantity less than or equal to 0).	The user submitted an item for an order with a negative or zero quantity.
	The catalog item(s) <CatalogItemIDs> has (have) been deleted by the provider. Please do not add the item(s) to the order.	The user tried to add a catalog item that has been deleted by the provider.
	The catalog item(s) <CatalogItemIDs> is (are) not permitted to order. Please do not add the item(s) to the order.	The user tried to add an order-restricted item to an order.
	There is no property <ChildOption> defined for the type: <ParentOption>	Error returned when an invalid option structure is associated with an item.
	Unable to get Catalog Item [<CatalogItemID>] because Invalid CatalogItem id: <CatalogItemID>.	The CatalogItemID submitted is not valid.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
CreateOrder	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.

Transaction	Error Message	Description
DeleteOrder	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
DeleteOrderLineItem	Unable to get Catalog Item [<CatalogItemID>] because Invalid CatalogItem id: <CatalogItemID>.	The CatalogItemID submitted is not valid.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
DeleteProviderOrder	Find Provider: (<ProviderID>) for ProviderOrder resulted in error.	The user submitted a ProviderID that does not exist.
	There is no property <ChildOption> defined for the type: <ParentOption>	The ProviderID submitted does not exist..
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
PresentCatalogItem	The catalog item(s) <CatalogItemIDs> has (have) been deleted by the provider.	A requested item has been deleted by the provider.
	The catalog item(s) <CatalogItemIDs> is(are) not permitted to order.	An item ordered is restricted to the user.
	Unable to find catalog item [<catalogItemIDs>]	The user tried to request information for an item that does not exist.
PresentOptionDefinitionsForOrder	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unable to locate order [<OrderID>].	The OrderID submitted is not valid for the provider.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
PresentOptionDefinitionsFor-ProviderOrder	Find Provider: (<ProviderID>) for ProviderOrder resulted in error.	The user submitted a ProviderID that does not exist.
	There is no property <ChildOption> defined for the type: <ParentOption>	The ProviderID submitted does not exist..
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
PresentOrder	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
PresentProviderOrder	Find Provider: (<ProviderID>) for ProviderOrder resulted in error.	The user submitted a ProviderID that does not exist.
	There is no property <ChildOption> defined for the type: <ParentOption>	The ProviderID submitted does not exist..

Transaction	Error Message	Description
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
QuoteOrder	Order must be validated before quoting.	The user submitted a request for an order quote before validating the order.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
SetOptionSelectionsForOrder	"[<OptionName>] is not a valid property for this item	The user tried to access an option that is not valid for an item.
	"[<OptionName>] is not a valid property for this item	The user submitted an invalid option structure.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
SetOptionSelectionsForProvider-Order	Find Provider: (<ProviderID>) for ProviderOrder resulted in error.	The user submitted a ProviderID that does not exist.
	"[<OptionName>] is not a valid property for this item	The user tried to access an option that is not valid for an item.
	There is no property <ChildOption> defined for the type: <ParentOption>	The user submitted an invalid option structure.
	There is no property <ChildOption> defined for the type: <ParentOption>	The ProviderID submitted does not exist..
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
SetUserInformation ForOrder	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
SubmitOrder	Order must be validated or quoted before submitting.	The user submitted an order before validating the order.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
UpdateOrderLineItem	"[<OptionName>] is not a valid property for this item	An invalid option is associated with an item

Transaction	Error Message	Description
	gov.nasa.echo.business.ejb.order. OrderException (Cannot add an item with quantity less than or equal to 0).	The user submitted an item for an order with a negative or zero quantity.
	The catalog item(s) <CatalogItemIDs> has (have) been deleted by the provider. Please do not add the item(s) to the order.	The user tried to add a catalog item that has been deleted by the provider.
	The catalog item(s) <CatalogItemIDs> is (are) not permitted to order. Please do not add the item(s) to the order.	The user tried to add an order-restricted item to an order.
	There is no property <ChildOption> defined for the type: <ParentOption>	Error returned when an invalid option structure is associated with an item.
	Unable to get Catalog Item [<CatalogItemID>] because Invalid CatalogItem id: <CatalogItemID>.	The CatalogItemID submitted is not valid.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
ValidateOrder	No shipping address specified.	The user submitted an order without the required shipping address information.
	Order options not defined for item [<CatalogItemID>].	The user submitted an order without the required options.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.

6.3.6 *Order State Conversions*

Order states are based on the provider order state. An order consists of zero or many provider orders. An order state is calculated from provider order state(s). The table below shows the states for orders made up of zero or one provider order, or multiple provider orders having the same provider order state.

Table 15: Order State Conversion: Zero or One Provider Order, or Multiple Provider Orders having the same provider order state.

# of Provider Orders	Provider Order State	Order State
0	--	NOT_VALIDATED
1	--	Same as provider order state
	QUOTE_FAILED	QUOTED_WITH_EXCEPTIONS

	QUOTE_REJECTED	
	SUBMIT_FAILED	SUBMITTED_WITH_EXCEPTIONS
	SUBMIT_REJECTED	

The figure below shows how to determine the state of an order made up of more than one provider order.

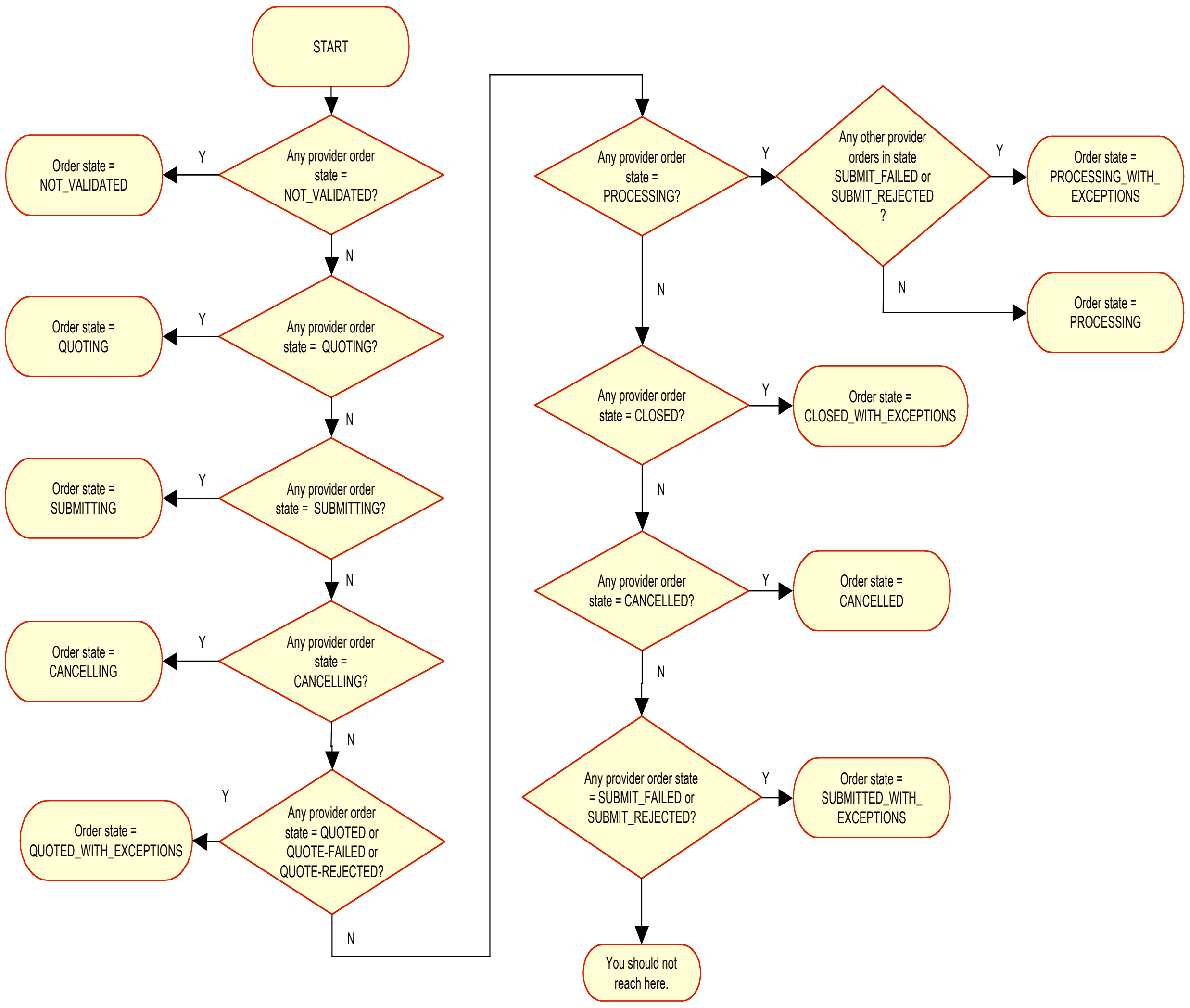

START
Any provider order state = NOT_VALIDATED?
Y
Order state = NOT_VALIDATED
N
Any provider order state = QUOTING?
Y
Order state = QUOTING
N
Any provider order state = SUBMITTING?
Y
Order state = SUBMITTING
N
Any provider order state = CANCELLING?
Y
Order state = CANCELLING
N
Any provider order state = QUOTED or QUOTE-FAILED or QUOTE-REJECTED?
Y
Order state = QUOTED_WITH_EXCEPTIONS
N
Any provider order state = PROCESSING?
Y
Any other provider orders in state SUBMIT_FAILED or SUBMIT_REJECTED ?
Y
Order state = PROCESSING_WITH_ EXCEPTIONS
N
Order state = PROCESSING
N
Any provider order state = CLOSED?
Y
Order state = CLOSED_WITH_EXCEPTIONS
N
Any provider order state = CANCELLED?
Y
Order state = CANCELLED
N
Any provider order state = SUBMIT_FAILED or SUBMIT_REJECTED?
Y
Order state = SUBMITTED_WITH_ EXCEPTIONS
You should not reach here.

Figure 11. Order state conversion: more than one provider order.

6.3.7 *Caveats in the Order Entry Service*

6.3.7.1 Restriction on order items

Some catalog items may only be ordered by those with permission. The restriction and permissions on catalog item ordering are set by the provider. Restrictions can apply to individual users, groups of users, or all users. If a catalog item is not available to a particular user and that user executes the PresentCatalogItem, CreateOrder, or AddOrderLineItem transaction including this catalog item, a message will show to the user that it is not an orderable item. If a provider changes the restriction/permission on an once orderable item to not orderable after the order has been created, the user who has the item in his/her order will be asked to remove this item from the order when the user UpdateOrderLineItem, ValidateOrder, QuoteOrder, or SubmitOrder.

6.3.7.2 Deleted Items

Providers can delete their data items. ECHO may have records for these items, but they can no longer be ordered. If a user tries to PresentCatalogItem, CreateOrder, or AddOrderLineItem including the deleted catalog item, an error message will be returned, indicating that it is a deleted item. If a provider deletes an item after the order has been created, the user who has the item in their order will be asked to remove this item from the order when executing the UpdateOrderLineItem, ValidateOrder, QuoteOrder, or SubmitOrder transaction.

6.4 Subscription Service

6.4.1 Transactions

The Subscription Service allows ECHO users to subscribe to metadata from different providers. When providers update the metadata repository, ECHO delivers the updated metadata to each subscriber. Although this capability is useful for end users, it can be extremely valuable for client developers, because it provides an extension point from ECHO. A subscription can be created to deliver new metadata to a client infrastructure that performs post-processing and data massaging in order to convert the data into a more usable form.

The three keys in creating a subscription are:

1. **What dataset are you interested in?** To determine what providers exist in the system, you can use the ListAllProviders transaction in the Provider Profile Service. Additionally, the Catalog Service discovery mechanism retrieves the datasets currently stored in the metadata repository. For more information, see the documentation for the respective services.
2. **What type of metadata from the dataset are you interested in?** ECHO allows a user to subscribe to collection level metadata, granule metadata, or both. Collection metadata will rarely change whereas granule metadata will frequently be updated or created. Most users choose to subscribe to granule metadata for this reason. Granule metadata may be deleted by the provider from time to time. If this happens, users will receive a notification email from ECHO about the granule deletion.
3. **Where should ECHO send the metadata when it is updated?** ECHO supports two methods of subscription delivery: FTP and e-mail. Due to security restrictions, we require all FTP deliveries to be transferred via passive-mode. If delivery cannot be made (because the FTP server is full or the mail server rejected the large file attachment), ECHO sends a notification e-mail to the subscriber. Consult your local administrator for more information on passive-mode transfers and file size limitations.

The Subscription Service supports a full set of transactions that allow the subscribers to easily manage their subscriptions. The following sections detail the transactions in metadata subscription service.

6.4.1.1 CreateSubscription

To receive ECHO metadata updates, the first step is creating a metadata subscription by sending CreateSubscriptionRequest message to ECHO.

Code Example 158. Metadata subscription

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <CreateSubscriptionRequest>
      <MetadataSubscription>
         <SubscriptionName>myFirstSubscription</SubscriptionName>
         <providerName>ORNL_TS1</providerName>
         <datasetName>MODIS Geolocation V001</datasetName>
```

```
        <MetadataFilterInfo>
                <spatialCondition>GLOBAL</spatialCondition>
        </MetadataFilterInfo>

        <MetadataActionInfo>
           <PresentationDescription>
              <DTDType><ECHO/></DTDType>
           </PresentationDescription>

           <CompressionType><GZIP/></CompressionType>

           <SubscriptionUpdateType><BOTH/></SubscriptionUpdateType>
        </MetadataActionInfo>

        <DeliveryInfo>
           <DeliveryType>   <FTPPUSH/></DeliveryType>
           <deliveryAddress>anonymous@ftp.company.com</deliveryAddress>
           <password>passwordvalue</password>
         <deliveryFolder>/pub/incoming/echo</deliveryFolder>
           <limitSize>4.0</limitSize>
        </DeliveryInfo>

        <StopTime>2003-09-30</StopTime>

     </MetadataSubscription>
  </CreateSubscriptionRequest>
</SubscriptionService>
```

6.4.1.1.1 SubscriptionName

A unique name given to each subscription. No two subscriptions belonging to the same user may have the same name.

6.4.1.1.2 ProviderName

The UserID of the provider whose data we are interested in.

6.4.1.1.3 DataSetName

The specific dataset we are interested in. This can be discovered through the Catalog Service.

6.4.1.1.4 MetaDataFilterInfo

Contains the spatial constraint to use for the subscription. This allows a user to narrow the data they are interested into a specific geographic location. *NOTE: only "GLOBAL" is an appropriate value to use. The ability to refine subscriptions to smaller spatial regions is not yet implemented.*

6.4.1.1.5 MetaDataActionInfo

Contains the information related to processing of the subscription. The PresentationDescription allows a user to specify a DTD format to which the XML sent to the client must conform. Users can also put their interested attribute names in the TupleType list. For example, if a user is only interested in the "platform" metadata, they can use the TupleType parameter to indicate this. In the above example, the user wants to receive data in "ECHO" format, with all available attributes to be presented. The TupleType validation is based on collection DTDs if the subscription update type is ALL_COLLECTIONS or COLLECTIONS_ONLY. The TupleType validation is based on granule DTDs if the subscription update type is GRANULES_ONLY or BOTH.

6.4.1.1.6 CompressionType

Allows users to specify any compression they desire on the delivered metadata file. It should be noted that the subscription files delivered from ECHO are XML formatted and hence highly compressible. It is highly recommended that users use a CompressionType of GZIP to conserve space and bandwidth.

6.4.1.1.7 SubscriptionUpdateType

Allows the user to subscribe to four types of metadata:

1. All Collection Metadata: The subscribers will receive every updated collection metadata. This type refers to ALL_COLLECTIONS for SubscriptionUpdateType.
2. A Specific Collection's Collection Metadata: The subscribers will only receive updated collection metadata of a subscriber-specified collection. This type refers to COLLECTIONS_ONLY for SubscriptionUpdateType.
3. A Specific Collection's Granule Metadata: The subscriber will receive every updated granule metadata in a subscriber-specified collection. This type refers to GRANULES_ONLY for SubscriptionUpdateType.
4. A Specific Collection's Granule and Collection Metadata: The subscriber will receive both the updated collection metadata and every updated granule metadata in a subscriber-specified collection. This type refers to BOTH for SubscriptionUpdateType.

6.4.1.1.8 DeliveryInfo

The Subscription Service allows the user to specify how metadata should be delivered to them. Currently, ECHO supports E-mail and FTP as delivery mechanisms. If E-mail is chosen, the deliveryAddress should be of the format "username@domain". If FTP Push is chosen, the deliveryAddress should also be of the format "username@ftp.domain". For FTP deliveries, the deliveryFolder and Password fields are required. In the above example, the user wants the metadata to be delivered via FTP push to the server ftp.domain, with the user "anonymous" with password "passwordvalue". The metadata sent from ECHO will be placed in the delivery folder "/pub/incoming/echo".

The Subscription Service has a governor that prevents delivery of metadata above a certain threshold. The limitSize is the max size (in megabytes) of the delivery file that the user will accept. The size of the data file is calculated prior to delivery. If the size of the data file exceeds the user specified limitSize, the system does not deliver the file. In addition, the user's email address in the user's profile is used in order that information about their subscription can be sent to them (such as notices when it expires, or if the file to be sent exceeds the set limitation on size). Also, after the data is uploaded to the registered ftp site (if FTP Push is selected), notice is also sent to the user at their e-mail address.

6.4.1.1.9 StopTime.

Each subscription has an expiration time that is specified in the StopTime. This field allows a user to specify a time after which no metadata should be delivered. In this example, the subscription will expire on September 30, 2003.

6.4.1.2 DeleteSubscription

Code Example 159. Deleting a subscription

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <DeleteSubscriptionRequest>
      <SubscriptionName>myFirstSubscription</SubscriptionName>
   </DeleteSubscriptionRequest>
</SubscriptionService>
```

6.4.1.3 ListSubscriptionNames

To enable management of multiple subscriptions, the user can use ListSubscriptionNames transaction to list all active subscription names.

Code Example 160. Listing active subscriptions

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <ListSubscriptionNamesRequest/>
</SubscriptionService>
```

6.4.1.4 PauseSubscription

This transaction will pause an active metadata subscription. No metadata updates will be received for a paused subscription. A paused subscription can be resumed using ResumeSubscription transaction.

Code Example 161. Pausing an active subscription

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <PauseSubscriptionRequest>
      <SubscriptionName>myFirstSubscription</SubscriptionName>
   </PauseSubscriptionRequest>
</SubscriptionService>
```

6.4.1.5 PresentSubscription

Code Example 162. Presenting a subscription

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <PresentSubscriptionRequest>
      <SubscriptionName>myFirstSubscription</SubscriptionName>
   </PresentSubscriptionRequest>
</SubscriptionService>
```

6.4.1.6 ResumeSubscription

Code Example 163. Resuming a paused subscription

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <ResumeSubscriptionRequest>
      <SubscriptionName>myFirstSubscription</SubscriptionName>
   </ResumeSubscriptionRequest>
```

```
</SubscriptionService>
```

6.4.1.7 UpdateSubscription

This transaction updates the information for an existing metadata subscription. Active, paused, and expired subscriptions can all be updated.

6.4.1.8 ListExpiredSubscriptionNames

This transaction generates a list of the names of all of the user's expired metadata subscriptions. The response includes an indication of whether the action succeeded, as well as the names of all expired metadata subscriptions associated with the user.

6.4.1.9 ListPausedSubscriptionNames

This transaction generates a list of the names of all of the user's paused metadata subscriptions. The response includes an indication of whether the action succeeded, as well as the names of all paused metadata subscriptions associated with the user.

6.4.1.10 RenewSubscription

A subscription expires on its stop date, which is identified in the CreateSubscription request. After this date, use this transaction to renew an expired metadata subscription, causing it to become active again. The request message identifies the subscription name and the new stop date. The response indicates whether the action succeeded.

6.4.2 Subscription Service and Restricted Data

Prior to delivering metadata to subscribers, the Subscription Service honors the visibility restrictions as defined by the access control lists in the Data Management Service. If a granule or collection is updated, but restrictions prevent viewing, the granule or collection will not be delivered to the end user. Moreover, no indication is given that the granule or collection was updated but not delivered to the user because of the restriction.

A current limitation of the Subscription Service is that previously updated restricted metadata is not delivered to the user when the data becomes visible. This "feature" is witnessed under the following condition:

- (As a user) Create a subscription to collection MOD01
- (As a provider) Create a Rolling Temporal Restriction against MOD01 during Sept 1 2002 – Sept 30 2002
- (As a provider) Update the MOD01 collection on Sept 15 2002

Under the above condition, the user will not receive the changes sent on Sept 15, 2002. Furthermore, the user will not receive the changes even after the restriction expires on Sept 30, 2002. Updates to MOD01 that occur post-Sept 30, 2002 will be sent to the user. This limitation is being investigated and may be overcome in the near future.

6.4.3 Subscription Service Error Messages

Three types of errors can occur when sending a valid request to the Subscription Service:

1. Incorrect subscription status transition: For example, if a user sends a ResumeSubscription request to resume an active subscription, this transaction cannot be granted. ECHO first finds out that the subscription is already in active state, and then includes an error message "The metadatasubscription myFirstSubscription has status active. Cannot perform ResumeMetadata Transaction. Its status must be paused to be resumed" in the response message.
2. Incorrect subscription name: If the user gives an incorrect subscription name – one that either does not belong to this user or does not exist in the ECHO datastore – ECHO generates an error "The user John has no subscription named myFirstSubscription" in the response message.
3. Required fields are empty: In some request messages, even valid XML documents, a required field is left empty. For example, in the CreateSubscription request, the subscription name may be left blank. In this case, ECHO generates an error "SubscriptionName is not valid" in the response message.

The following table gives the possible error messages associated with each transaction within the Subscription Service.

Table 16: Subscription service error messages.

Transaction	Error Message	Description
CreateSubscription	<providerName> is an existing ECHO provider. Please choose a provider from the list of ECHO providers:: <providerName1>,<providerName2>…	This error is returned when the provider name requested does not exist. There is a typo in the message. The correct error message should be "<providerName> is not an existing ECHO provider. Please choose a provider from the list of ECHO providers:: <providerName1>,<providerName2>…"
	<datasetName> is not available from the <providerName> provider	This error is returned when the requested dataset of the provider indicated does not exist.
	DeliveryAddress is invalid. Please check its format. The expected format for the DeliveryAddress is username@domain	This error is returned when delivery email address is invalid meaning the delivery email address does not conform to the format of username@domain.
	Password cannot be null. You need to provide a password for a FTPPUSH delivery.	This error is returned when FTP push was chosen as delivery method but the password for login to the destination is not provided.
	DeliveryFolder cannot be null. You need to provide a DeliveryFolder for a FTPPUSH delivery.	This error is returned when FTP push was chosen as delivery method but the delivery folder at the destination is not provided.
	DeliveryLimit Size is a float measured in Megabytes. Please check its format.	This error is returned when size limitation on deliverable is not expressed as a number.
	StopTime cannot be before today.	This error is returned when the time for stop subscription indicated is earlier then today's date.
	SubscriptionName: <SubscriptionName> already exists for the user <UserName>.	This error is returned when submit a subscription using a subscription name that already exists for the user.
UpdateSubscription PresentSubscription DeleteSubscription PauseSubscription ResumeSubscription	The user <UserName> has no subscription named <SubscriptionName>.	This error is returned when a user trying to access a subscription that does not exist for the user. All the error messages listed for CreateSubscription except the last one listed apply.
CreateSubscription PresentSubscription DeleteSubscription ListSubscriptionNames PauseSubscription ResumeSubscription	[SessionManager] The service {ProviderOrderManagementService} that you attempted to access either does not exist or you do not have the appropriate access level to use it.	This error is returned when trying to access this transaction by a user who does not have the privilege for this transaction.
PauseSubscription ResumeSubscription	The metadatasubscription -783369493 has status P. Cannot perform PauseMetadata Transaction. Its	This error is returned when a user trying to pause a subscription that is already been

	status must be active to be paused.	paused.

6.5 ECHO JAVA Façade

6.5.1 Façade Introduction

ECHO uses an XML over http RPC style mechanism similar to SOAP and xmlrpc. The primary difference between ECHO and SOAP or xmlrpc is that ECHO has defined a set of DTDs that represent implemented operations. Making use of ECHO requires a client developer to marshal information to and from XML format and then pass the serialized request over a SOAP channel. A significant amount of overhead is incurred by the client developer in order to invoke a remote operation and interpret a response. The Façade has been developed to alleviate some of the burdens of the client developers.

6.5.2 XML Example

Assume a developer is implementing a client to the UserAccountService, allowing a user to update their address and telephone information. After the developer collects all relevant address information, they would have to assemble an XML message to send to ECHO. Below is an example of the amount of code required to generate the XML request to add an address to a user's profile:

```
Document requestDoc = new DocumentImpl();
Element service = requestDoc.createElement("UserAccountService");
requestDoc.appendChild(service);

Element transaction = requestDoc.createElement("AddAddressRequest");
service.appendChild(transaction);

Element addressInformationElement = requestDoc.createElement("AddressInformation");
transaction.appendChild(addressInformationElement);

Element idElement = requestDoc.createElement("AddressId");
addressInformationElement.appendChild(idElement);

Element theIdElement = requestDoc.createElement("AddressId");
theIdElement.appendChild(requestDoc.createTextNode(addressId));
IdElement.appendChild(theIdElement);

Element usFormatElement = requestDoc.createElement("USFormat");
usFormatElement.appendChild(requestDoc.createTextNode(usFormat));
addressInformationElement.appendChild(usFormatElement);

//  … and so on for Street1-5, City, State, and Zip Code

// submit the message and get the response
OutputFormat format = new OutputFormat(requestDoc);
String dtdAddress = http://api.echo.nasa.gov/dtd/UserAccountService.dtd;
format.setDoctype(null, dtdAddress);

StringWriter stringWriter = new StringWriter();
XMLSerializer serializer = new XMLSerializer(stringWriter, format);
serializer.asDOMSerializer();
serializer.serialize(requestDoc.getDocumentElement());

String requestMessage = stringWriter.toString();
String responseMessage = null;
try {
  responseMessage = echoToken.perform(requestMessage);
} catch (XMLServiceException e) {
  // do something intelligent
}

// now parse the response message
```

The above code demonstrates how a developer assembles an XML message and submits it to ECHO. It is a relatively simple and straightforward example because it does not contain any hierarchical nested complex structures. Assembling and submitting XML that interacts with the Options framework within ECHO is extremely difficult.

6.5.3 Façade Example

In this example, the developer leverages the services of the Façade to provide XML message assembly and invocation. Using the Façade allows the client developer to focus on providing business level functionality and not data marshalling and remote invocation services.

```
AddressInformationDO address = null;
try {
  address = new AddressInformationDO(new AddressID("home"), true, street1, street2, street3,
                                                              street4, street5, city, state, zip, country);
} catch (ValidationException e) {
  // some validation can be performed client side within remote invocation
}

try {
  UserAccountService.AddAddress(address, echoToken);
} catch (XMLServiceException e) {
  // some validation can only be performed on the server, this is where those errors are reported
}
```

6.5.4 Façade Limitations

The Façade only provides data marshalling for first-class parameters and enumerated types. It does not support data serialization and deserialization of queries and their responses. If you are developing a client that queries ECHO, you can (and should) use the Façade for all parameters you pass to the Query transaction except for the actual AQL query. The actual query still needs to be created using a mechanism external to the Façade, but it is still recommended that you use the Façade for assembling all other XML structures external to the query itself.

6.5.5 Façade Applicability

The Façade manages Java to XML translations (and XML to Java translations), and is both stateless and platform independent. A Façade is released for each version of ECHO, but there is nothing that prevents a client developer from using version 4.8 of the Façade to connect to version 6.0 of the ECHO server side software. If the APIs invoked have changed, the Façade will not be able to perform the remote invocation. So long as the APIs are the same, the Façade can talk to any ECHO version or platform.

7 Data Partner Interface

Through the use of API's set-up by the ECHO Development team, Data Partners are provided a variety of services to control their metadata. Data Partner APIs can be found on the ECHO Web site. Complete metadata control includes the ability to query and view the very same metadata as seen by the user community. Data Partner APIs are described in the following sections.

7.1 Provider Account Service

The Provider Account Service allows registered providers to set policy information, maintain notification subscriptions and maintain basic organization and contact information.

Provider accounts exist in the system and are similar to user accounts. Provider accounts have user names, passwords, address information, and contact information. The primary difference between a Provider account and a normal user account is that a Provider account contains one or more Contact entities and a Provider account can control policy information for their catalog items.

7.1.1 Transactions

7.1.1.1 AddContact

The key difference between a user account and a provider account is the ability for a provider account to specify Contact entities. Contact entities are additional pieces of information that a Provider can associate with itself. For example, a Provider might have a Billing Department and a Customer Support Department. These two departments are good candidates for a Contact entity. Each Contact entity has a first and last name, address information, and e-mail address. Thus, a Provider that has geographically separate offices may capture and present that information to ECHO client users.

This transaction enables one or more provider contacts to be added for this ECHO provider. A contact is an individual who has a named role within the organization. It has role name, first name, last name, address information, phone information and email. Each contact is distinct by role name, which identifies this contact's role, like "shipping manager", "order manager".

If a contact named "order manager" is created, e-mails that are sent to users regarding orders will be "CC'd" to this contact's e-mail address.

Code Example 164. Add provider contact.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <AddContactRequest>
      <ProviderContact>
         <ContactRole>Billing Contact</ContactRole>
         <ContactFirstName>Bill</ContactFirstName>
         <ContactLastName>Manager</ContactLastName>
         <AddressInformation>
            <AddressID>Office</AddressID>
            <USFormat>TRUE</USFormat>
            <Street1>9111 Edmonston Rd</Street1>
            <City>Greenbelt</City>
            <State>MD</State>
            <Zip>20770</Zip>
            <Country>USA</Country>
         </AddressInformation>
         <PhoneInformation>
            <PhoneName>Office</PhoneName>
            <PhoneString>1-301-555-888</PhoneString>
         </PhoneInformation>
         <EMailAddress>bill@host.com</EMailAddress>
      </ProviderContact>
   </AddContactRequest>
</ProviderAccountService>
```

7.1.1.2 DeleteContact

This transaction enables provider contacts to be deleted. Contact roles have to be specified in order to delete.

Code Example 165. Delete provider contacts.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
    <DeleteContactRequest>
      <ContactRole>shipping contact</ContactRole>
    </DeleteContactRequest>
</ProviderAccountService>
```

7.1.1.3 PresentContacts

This transaction enables provider contacts information to be presented. If contact roles are specified, then only the specified contacts are presented. The contact information includes contact role, first name, last name, address information, phone information and e-mail.

Code Example 166. Present contacts (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <PresentContactsRequest/>
</ProviderAccountService>
```

Code Example 167. Present contacts (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <PresentContactsRequest>
      <ContactRole>billing contact</ContactRole>
   </PresentContactsRequest>
</ProviderAccountService>
```

7.1.1.4 PresentPolicyDefinitions

This transaction enables the provider policy definitions to be presented. Normally, the policy definitions are defined in the ECHO system, and they can be modified based on the needs of all ECHO providers.

Code Example 168. Present policy definitions.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <PresentPolicyDefinitionsRequest/>
</ProviderAccountService>
```

7.1.1.5 PresentPolicySelections

This transaction enables the provider policy selections to be presented. The provider may specify the policy name. If the policy name is not specified, the complete list of current policy selections will be presented.

Code Example 169. Present policy selections.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <PresentPolicySelectionsRequest/>
</ProviderAccountService>
```

7.1.1.6 Present Provider Info

This transaction enables the provider information and provider contacts information to be presented. Currently, the provider information just has provider’s organization information.

Code Example 170. Present provider information.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <PresentProviderInfoRequest/>
</ProviderAccountService>
```

7.1.1.7 SetPolicySelections

This transaction enables the policy selections for a registered provider to be set based on the policy definitions defined for the selections. The selection will be validated according to the policy definition. More information about options can be found in the Orders and Options section.

Code Example 171. Set Policy Selections

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <SetPolicySelectionsRequest>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>option1</OptionName>
            <Value>value1</Value>
         </SimpleOptionSelection>
      </OptionSelection>
   </SetPolicySelectionsRequest>
</ProviderAccountService>
```

7.1.1.8 UpdateContact

To be supplied.

7.1.1.9 UpdateProviderContacts

This transaction enables the provider contacts to be updated. The updating information includes first name, last name, address information, phone information and email. Contact role can't be updated.

Code Example 172. Update provider contacts.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <UpdateContactRequest>
      <ProviderContact>
         <ContactRole>billing contact</ContactRole>
         <ContactFirstName>AJ</ContactFirstName>
         <ContactLastName>Smith</ContactLastName>
      </ProviderContact>
   </UpdateContactRequest>
</ProviderAccountService>
```

7.1.1.10 UpdateProviderInformation

This transaction enables provider information, currently only the organization information, to be updated.

Code Example 173. Update provider information.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <UpdateProviderInformationRequest>
      <ProviderInformation>
         <OrganizationName>NASA_DAAC</OrganizationName>
      </ProviderInformation>
   </UpdateProviderInformationRequest>
</ProviderAccountService>
```

7.1.1.11 GrantProviderAccess

This transaction grants the specified users access to a provider. The user who grants access must currently have access to the provider they are interested in. Also, that user must either have set their provider context to that provider or they must only have access to just that one provider.

Code Example 174. Grant provider access.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <GrantProviderAccessRequest>
      <UserName>User101</UserName>
   </GrantProviderAccessRequest>
</ProviderAccountService>
```

7.1.1.12 RevokeProviderAccess

This transaction revokes access to a provider from the specified users. The user who revokes access must currently have access to the provider they are interested in. Also, that user must either have set their provider context to that provider or they must only have access to just that one provider.

Code Example 175. Revoke provider access.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <RevokeProviderAccessRequest>
      <UserName>User101</UserName>
   </RevokeProviderAccessRequest>
</ProviderAccountService>
```

7.1.1.13 Provider Inspect Functions

Inspect functions allow the provider to view information about their holdings stored in ECHO. Providers can use this information to validate their holdings. Since these transactions are for providers, ECHO imposes no visibility restrictions on the datasets (in this context, “dataset” is synonymous with “collection”) and granules.

Note: The ECHO system does not perform the validation. Providers validate their holdings based on the results returned from these transactions.

7.1.1.13.1 InspectHoldings

This transaction returns the following information:

- Names of the provider’s datasets
- Total number of granules in each dataset

7.1.1.13.2 InspectDataset

This transaction returns the following information:

- Granule UR
- Provider’s last update time. Time ranges may be specified based on the provider’s last update time
- ECHO’s last update time

Providers must select the method by which they would like to receive the results from this transaction:

- Inline
- E-mail
- FTP. ECHO returns an FTP URL for the provider to download

Notes:

ECHO imposes a 1,000,000 granule size limit on inline and e-mail results.

The FTP URL expires two (2) days after ECHO receives the request.

Please allow ECHO time to assemble the FTP file for large datasets.

The code examples below demonstrate how to select the method by which they receive results.

Code Example 176. Inspect dataset inline return.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <InspectDatasetRequest>
      <DataSetID>ASTER Digital Elevation Model V003</DataSetID>
      <ProviderLastUpdateRange>
         <StartTime>2001-01-10</StartTime>
         <StopTime>2002-01-10</StopTime>
      </ProviderLastUpdateRange>
      <InspectionReturnType>
         <INLINE/>
      </InspectionReturnType>
   </InspectDatasetRequest>
</ProviderAccountService>
```

Code Example 177. Inspect dataset e-mail return.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <InspectDatasetRequest>
      <DataSetID>ASTER Digital Elevation Model V003</DataSetID>
      <ProviderLastUpdateRange>
         <StartTime>2001-01-10</StartTime>
         <StopTime>2003-01-10</StopTime>
      </ProviderLastUpdateRange>
      <InspectionReturnType>
         <EMAIL/>
      </InspectionReturnType>
      <EmailString>wu@gst.com</EmailString>
   </InspectDatasetRequest>
</ProviderAccountService>
```

Code Example 178. Inspect dataset FTP return.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <InspectDatasetRequest>
      <DataSetID>ASTER Digital Elevation Model V003</DataSetID>
      <ProviderLastUpdateRange>
         <StartTime>2001-01-10</StartTime>
         <StopTime>2003-01-10</StopTime>
      </ProviderLastUpdateRange>
      <InspectionReturnType>
         <FTP/>
      </InspectionReturnType>
      <EmailString>wu@gst.com</EmailString>
   </InspectDatasetRequest>
</ProviderAccountService>
```

7.1.1.14 ListUsersWithProviderRole

This transaction lists all users who have the same provider role as the caller of this transaction. The user must have set provider context (If the user is an admin user or the user has multiple provider roles) before invoking this transaction.

Code Example 179. List users with provider role request

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService SYSTEM "http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
         <ListUsersWithProviderRoleRequest/>
</ProviderAccountService>
```

Code Example 180. List users with provider role response

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE ProviderAccountService SYSTEM "http://api.echo.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
         <ListUsersWithProviderRoleResponse>
                  <BooleanResult>
                           <BooleanResultType>
                                    <REQUEST_SUCCEEDED />
                           </BooleanResultType>
                  </BooleanResult>
                  <UserName>gsfc_user1</UserName>
                     <UserName>gsfcecs_rolemgr</UserName>
         </ListUsersWithProviderRoleResponse >
</ProviderAccountService>
```

7.1.2 Addresses

Like regular user accounts, provider accounts contain multiple address entities. A provider account may have separate address entities associated with customer support locations, POP locations, etc.

7.1.3 Emails

Provider accounts have special capabilities beyond those of a regular client user. A Provider has multiple contacts associated with his / her account. Each contact contains an email address. Therefore, a provider account may have multiple email addresses associated with it

7.2 Metadata Ingest

Metadata ingest is a backend process in the ECHO system. The main task for the metadata ingest process is loading bulk metadata and updating the database. To improve the performance and efficiency of input data handling, the metadata ingest process uses FTP as the data transmission interface for the data provider metadata submission. The metadata input includes collection, granule, and browse metadata. It must be in XML format and conform to the ECHO DTDs. The ECHO metadata DTD is published on the ECHO Web site at http://eos.nasa.gov/echo. If a data provider generates the XML file for data ingest using their own DTD, then the provider's DTD must be converted to ECHO's DTD before sending into ECHO.

Each data provider is assigned a database account to host his or her metadata. Data providers are referred to by their own provider name - a unique identity in the ECHO system. The provider name is determined by agreement between the data provider and the ECHO Operations team. When ingesting metadata, data are being inserted, deleted or updated in the database. A data provider ingest only affects that provider's metadata.

The following sections discuss ECHO DTDs defined for collection metadata input and granule metadata input.

7.2.1 Notification of Metadata Transmission

No specific notification of metadata transmission is required, but it is helpful to ECHO operations staff. The ECHO operations staff will monitor the providers' ftp repository for file transmission and completion. This monitoring may also be performed by the auto ingest process when running.

7.2.2 Spatial Information

The ECHO system uses the Oracle database to store the spatial information for collections and granules. It is important for data providers to prepare the spatial data appropriately to ensure that the spatial search will return correct results.

ECHO system accepts both Cartesian and Geodetic coordinate systems (WGS 84). A provider chooses a coordinate system based on the size and location of the spatial area covered.

With Oracle Cartesian coordinate system, the acceptable spatial data types include Point, Circle, Line, Bounding Box, and Polygon. With Oracle Geodetic coordinate system (WGS 84), the spatial data types supported include point, line, and polygon.

Table 17: Supported spatial data types.

Spatial data type	Cartesian	Geodetic	Notes
Point	✓	✓	
Circle	✓	×	Oracle spatial uses three points on the circumference of the circle. Any spatial data received as a circle expressed in center-latitude, center-longitude, and radius will not be stored as Oracle spatial data. Instead, it will be stored in a regular relational table.
Line	✓	✓	A line that has vertices across the International Date Line or across the poles will be considered invalid spatial data for flat Cartesian coordinate system.
Bounding box	✓	×	Since the Geodetic model does not support the bounding box type, ECHO system stores bounding box data as a four vertices polygon in flat Cartesian coordinate system for data process unification.
Polygon	✓	✓	Polygon vertices must be stored in order of vertex connection. The vertices should be sent in clockwise order. In addition to the order of the vertices, any consecutive vertices cannot have the same latitude and longitude, i.e., no repeating points. A line or a polygon that has vertices across the International Date Line or across the poles will be considered invalid spatial data for flat Cartesian coordinate system.

The Oracle spatial Geodetic model uses the shortest distance line to connect two vertices in order to construct the polygon area or the line. If there is not enough density for a set of vertices, then the line or the polygon area could be misinterpreted or the data could be considered invalid. Any polygon should not cover more than half of the earth.

To prepare the ECHO system to support polar search and Geodetic search, we store the spatial data in three possible types: point, line, and polygon. To avoid misinterpretation of a data provider's spatial data, the ECHO system will not manipulate any of the spatial input data. Data providers are responsible for the correctness and integrity of their spatial data. When spatial data is sent to ECHO, data providers must notify ECHO in advance as to what coordinate system is being used for the spatial data. Presumably one coordinate system will be used per data set. Data providers should chose an appropriate data type for their spatial data. Data providers should also provide spatial data with appropriate density if using the Geodetic model. If any spatial data input is considered invalid by Oracle spatial, ingest for that data record will fail.

By default, ECHO assumes the maximum spatial coverage area for any data provider is the whole Earth (-180 to 180 for longitude and –90 to 90 for latitude). The data provider should specify the maximum spatial coverage area otherwise.

The application must define the resolution for vertices' degree representation. If any two vertices' difference is less than the resolution, those two vertices will be considered identical, which might cause the spatial data become invalid for Oracle spatial. The data provider should provide the resolution for both latitude and longitude for their spatial data in advance.

Note: At this time, a spatial search is not allowed on a collection or granule that uses a circle spatial coverage representation.

7.2.3 Temporal Information

The ECHO system does not require any specific date format. However, ECHO does require data providers to use only one date format for all data that reflects day/time information, and that they use one of the conventional date formats. The data provider must notify the ECHO Operations team, in advance, which date format is to be used. The value of the temporal information such as collection's range – beginning date and range ending date etc. – must be GMT/UTC.

7.2.4 Additional Attributes

The ECHO system stores the additional (as known as product specific) attributes including name and value exactly the way they are received. The ECHO system does not do a unification or mapping of the additional attributes' name among the data providers. Searches against Additional Attribute name and value are case-insensitive.

7.2.5 Metadata Mapping/Ingest Process

Metadata input files generated using the ECHO DTD are accepted and processed directly by the ingest process. Metadata input files conforming to other DTDs require an XML to XML conversion before being submitted to the ECHO ingest process. The provider is strongly recommended to perform the conversion. Contact the ECHO Operations team for more information.

When the ECHO ingest process detects potential input files in a provider's FTP input directory, the ingest process will then examine and validate these files. Each file will be checked to see if it is a properly formed XML file by examining the first line of text, which must contain the <!xml....> declaration. If this line is not present, the input file will be rejected. Next, the ingest process will verify that the file is located in the appropriate directory. For example, if a granule XML file is placed in the collection FTP input directory the input file will be rejected. Finally, the ingest process will validate the input file with its corresponding DTD. If this validation fails, the input file will be rejected. Whenever an input file is rejected, an error will be recorded and sent to the provider via the pre-configured provider contact email address. The same information will be emailed to the ECHO OPS staff as well.

Once an input file has been successfully validated, the ECHO ingest process will load the data into the ECHO database and update the operational database tables. The data will also then be made available for public search and an ingest summary email will be sent to the provider via the pre-configured provider contact email address. The ingest summary information will be emailed to the ECHO Operations staff as well.

Additional constraints are applied at ingest to maintain granule data integrity. These constraints are:

- Collection metadata must be in the ECHO database before any of that granule's metadata associated with the collection is accepted in ECHO.
- Granules' Additional Attributes must be a subset of their associated collection's Additional Attributes.
- Granules' platform/instrument/sensor configurations must be a subset of its associated collection's platform/instrument/sensor configurations.
- Granules' instrument operation modes must be a subset of its associated collection's instrument operation modes.
- Granules' analysis sources must be a subset of its associated collection's analysis sources.
- Granules' campaigns must be a subset of its associated collection's campaigns.
- A granule will be rejected if any of the above constraints are violated.

7.2.6 Translation Process

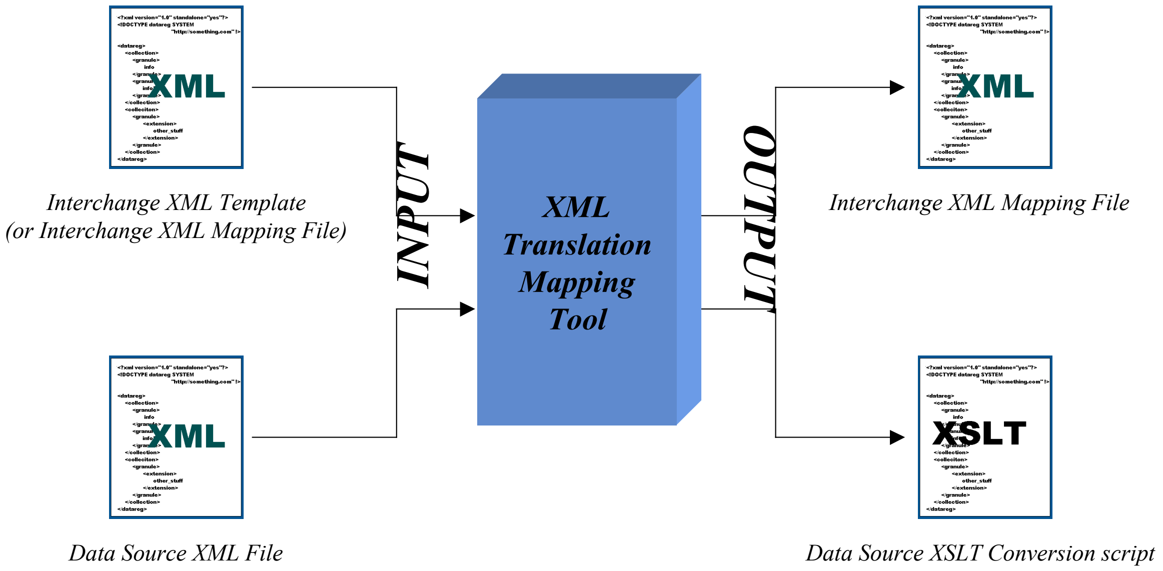

XML
Interchange XML Template
(or Interchange XML Mapping File)
XML
Data Source XML File
INPUT
XML Translation Mapping Tool
OUTPUT
XML
Interchange XML Mapping File
XSLT
Data Source XSLT Conversion script

Figure 12. XML Translation Process

Using a wizard-like interface, the XML Translation Mapping Tool helps you create complex mappings between the elements in your Data Source XML file and the relevant elements in the Interchange XML Template file.

7.2.7 Creating ECHO Compatible XML Files

ECHO has its own DTDs defined for collection and granule metadata. ECHO ingest processes XML metadata files compliant to the ECHO DTDs. Data providers should generate their metadata XML file using the ECHO DTDs.

7.2.7.1 ECHO Collection DTD

The complete ECHO Collection DTD is available from the ECHO Web site at http://eos.nasa.gov/echo.

7.2.7.1.1 Basic Collection Information

The required elements for collection metadata are ShortName, VersionID, InsertTime, LastUpdate, LongName, DataSetID, and CollectionDescription. Additional elements include VersionDescription, RevisionDate, SuggestedUsage1, SuggestedUsage2, ProcessingCenter, ArchiveCenter, CitationforExternalPublication, CollectionState, MaintenanceandUpdateFrequency and ProcessingLevelId. These can also help describe a provider's collection metadata for use in data searches.

The ShortName and version number VersionID combination uniquely identifies a collection. In addition to the ShortName and VersionID, ECHO also requires a dataset ID (DataSetID) to uniquely identify a collection for the provider, which is set by the data provider.

The InsertTime is the day and time the collection metadata was inserted in the provider's database. The LastUpdate is the day and time the collection metadata was last updated in the provider's database.

7.2.7.1.2 Temporal

Temporal information represents the time that the collection's data were collected. Three types of temporal information expression are used to present the temporal information associated with the collection. The types are: Single day time, Range day time, and Periodic day time. A collection could have more then one type of temporal information associated. This information is essential search criteria for collection.

Expression of temporal information (range day time).

Code Example 181.

```
<Temporal>
   <TimeType>UTC </TimeType>
   <DateType>Gregorian </DateType>
   <TemporalRangeType>Continuous Range</TemporalRangeType>
   <PrecisionofSeconds>1</PrecisionofSeconds>
   <EndsatPresentFlag>Y</EndsatPresentFlag>
   <RangeDateTime>
      <RangeBeginningDate>1998-01-01 00:00:00.0</RangeBeginningDate>
      <RangeBeginningTime>00:00:00.000000</RangeBeginningTime>
      <RangeEndingDate>1998-01-01 00:00:00.0</RangeEndingDate>
      <RangeEndingTime>00:00:00.000000</RangeEndingTime>
   </RangeDateTime>
</Temporal>
```

Temporal information must be submitted using UTC time and Gregorian day type.

The format of the date expression in this example is: yyyy-mm-dd hh24:mi:ss.ms(1) – four digits for year, two digits for month, two digits for day, two digits for hour in 24 hour system, two digits for minute, and two digits for second with one digit for precision of the second. Providers might express their date information in a different format and ECHO will process the information accordingly. However, once the format is defined and agreed upon between ECHO and the data provider, then all the date information coming from the data provider should follow the format. Otherwise, ECHO will not be able to process the information and will result in a NULL entry for the information.

7.2.7.1.3 Spatial

Spatial data is the spatial area that the collection's data covers. This is an essential piece of information for collection search criteria. For detailed information on handling and expression of spatial information, please refer to the Spatial Information section.

7.2.7.1.4 Sources and Sensors

ECHO adopts a layered representation of sources and sensors information for the collection. A source and sensor layer is defined as:

Platform->* Instrument->* Sensor

A collection could be associated with 0 or more platforms; each platform could contain 0 or more instruments, and each instrument could contain 0 or more sensors. There might be characteristic parameters associated with the platforms, instruments, and or sensors for the collection. In addition to the characteristic parameters, there might be an operational mode associated with instruments. When the data provider does not have the platform concept with their collection, the input for platform short name should be "No Platform Associated". When the data provider does not have the instrument concept for their collection, the input for instrument short name should be “No Instrument Associated”.

Full platform/instrument/sensor description.

```
<Platform>
   <PlatformShortName>Terra</PlatformShortName>
   <Instrument>
      <InstrumentShortName>MODIS</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
      <OperationMode>Operation Mode</OperationMode>
   </Instrument>
</Platform>
<Platform>
   <PlatformShortName>Terra</PlatformShortName>
   <PlatformLongName>First EOS Polar Orbiting Satellite, 10:30 AM Descending Equator Crossing</PlatformLongName>
   <PlatformType>Spacecraft</PlatformType>
   <PlatformCharacteristic>
      <PlatformCharacteristicName>EquatorCrossingTime</PlatformCharacteristicName>
      <PlatformCharacteristicDescription>Local time of the equator crossing and direction (ascending or
descending)</PlatformCharacteristicDescription>
      <PlatformCharacteristicDataType>varchar </PlatformCharacteristicDataType>
      <PlatformCharacteristicUnit>Local Mean Time</PlatformCharacteristicUnit>
      <PlatformCharacteristicValue>10:30, descending</PlatformCharacteristicValue>
   </PlatformCharacteristic>
   <Instrument>
      <InstrumentShortName>MODIS</InstrumentShortName>
      <InstrumentLongName>Moderate-Resolution Imaging Spectroradiometer</InstrumentLongName>
      <InstrumentTechnique>Imaging Spectroradiometry</InstrumentTechnique>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 182. No platform association.

```
<Platform>
   <PlatformShortName>No Platform Associated</PlatformShortName>
   <Instrument>
      <InstrumentShortName>MODIS</InstrumentShortName>
      <InstrumentLongName>Moderate-Resolution Imaging Spectroradiometer</InstrumentLongName>
      <InstrumentTechnique>Imaging Spectroradiometry</InstrumentTechnique>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 183. No instrument association.

```
<Platform>
   <PlatformShortName>Terra</PlatformShortName>
   <PlatformLongName>First EOS Polar Orbiting Satellite, 10:30 AM Descending Equator Crossing</PlatformLongName>
   <PlatformType>Spacecraft</PlatformType>
   <PlatformCharacteristic>
      <PlatformCharacteristicName>EquatorCrossingTime</PlatformCharacteristicName>
      <PlatformCharacteristicDescription>Local time of the equator crossing and direction (ascending or
descending)</PlatformCharacteristicDescription>
      <PlatformCharacteristicDataType>varchar </PlatformCharacteristicDataType>
      <PlatformCharacteristicUnit>Local Mean Time</PlatformCharacteristicUnit>
      <PlatformCharacteristicValue>10:30, descending</PlatformCharacteristicValue>
   </PlatformCharacteristic>
   <Instrument>
      <InstrumentShortName>No Instrument Associated</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 184. Sensor only.

```
<Platform>
   <PlatformShortName>No Platform Associated</PlatformShortName>
   <Instrument>
      <InstrumentShortName>No Instrument Associated</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

7.2.7.1.5 Keywords

ECHO supports three kinds of keyword associations for collections: discipline keywords, spatial keywords, and temporal keywords. For discipline keywords and spatial keywords, the requirement for keywords entry is compliant with GCMD keywords standard. For GCMD keywords standard, please see http://gcmd.gsfc.nasa.gov/Resources/valids/index.html. ECHO also supports discipline keywords association at a collection's instrument level.

7.2.7.1.6 Additional Attributes

Additional Attributes are parameters that further describe the collection. These are important search criteria for the collection. ECHO supports collection-level Additional Attributes definition. No Additional Attribute values are associated with a collection at this time.

Code Example 185. Collection-level Additional Attributes

```
<AdditionalAttributes>
   <AdditionalAttributeDataType>varchar </AdditionalAttributeDataType>
   <AdditionalAttributeDescription>Version of the process software that generated the
product.</AdditionalAttributeDescription>
   <AdditionalAttributeName>PROCESSVERSION</AdditionalAttributeName>
</AdditionalAttributes>

<AdditionalAttributes>
   <AdditionalAttributeDataType>String</AdditionalAttributeDataType>
   <AdditionalAttributeDescription>Estimated RMS error in geolocation</AdditionalAttributeDescription>
   <AdditionalAttributeName>GEO_EST_RMS_ERROR</AdditionalAttributeName>
</AdditionalAttributes>

<AdditionalAttributes>
   <AdditionalAttributeDataType>String</AdditionalAttributeDataType>
   <AdditionalAttributeDescription>Flag set (to 0) if science_state, the L1A engineering data flag that indicates the
Normal/Test configuration of the MODIS instrument, was set for at least one scan in the
granule.</AdditionalAttributeDescription>
   <AdditionalAttributeName>SCI_STATE</AdditionalAttributeName>
</AdditionalAttributes>

<AdditionalAttributes>
   <AdditionalAttributeDataType>String</AdditionalAttributeDataType>
   <AdditionalAttributeDescription>Flag set (to 0) if science_abnormal, the L1A engineering data ground-set flag that
indicates potentially abnormal science data due to things other than MODIS (such as maneuvers, data link, etc.), was set
for at least one scan in the granule.</AdditionalAttributeDescription>
   <AdditionalAttributeName>SCI_ABNORM</AdditionalAttributeName>
</AdditionalAttributes>

<AdditionalAttributes>
   <AdditionalAttributeDataType>int </AdditionalAttributeDataType>
   <AdditionalAttributeDescription>The number of the granule for the day starting at midnight. Example: The granule from
00:00:00 to 00:00:05 will be granule number 1. The granule from 00:01:00 to 00:01:05 will be granule number
13</AdditionalAttributeDescription>
   <AdditionalAttributeName>GRANULENUMBER</AdditionalAttributeName>
</AdditionalAttributes>
```

7.2.7.1.7 Other Descriptive Information

Other information associated with a collection includes:

- Computer Science Data Type (CSDT)
- Contact information
- Campaign information
- Association
- Browse
- Online URL

- Coordinate system and planar information for collection's raw data
- DIF Ids

7.2.7.1.8 Restriction Flag

ECHO will restrict collections from viewing and ordering by all except the user bearing the provider role by verifying collection's visibility flag. By setting the visibility flag to be restricted when ingest a new collection, ECHO will automatically restrict the new collection from viewing and ordering by all except the user bearing provider's role. This default restriction flag used for all granules in the collection can be overridden by setting restriction flag on a per granule basis.

Setting default restriction flag for a collection

Code Example 186. Restriction Flag for a Collection

```
          <RestrictionFlag>7</RestrictionFlag>
 <RestrictionComment>default for collection</RestrictionComment>
```

7.2.7.2 ECHO Granule DTD

The complete ECHO granule DTD is published on the ECHO Web site at http://eos.nasa.gov/echo.

7.2.7.2.1 Granule/Collection Association

Earth Science metadata represented by ECHO are based on granules. Every granule has to belong to a collection. A collection could contain 0 or more granules. The collection/granule association information is provided via granule metadata input.

Code Example 187. Granule/Collection Association

```
<GranuleURMetaData>
   <GranuleUR>SC:MODMGGAD.001:81677</GranuleUR>
   <DbID>81677</DbID>
   <InsertTime>2001-09-20 11:43:31.996</InsertTime>
   <LastUpdate>2001-09-20 11:43:31.996</LastUpdate>
   <CollectionMetaData>
      <ShortName>MODMGGAD</ShortName>
      <VersionID>1</VersionID>
   </CollectionMetaData>
……
</GranuleURMetaData>
```

A collection's short name and version ID, combined, uniquely identify the collection to which the granule belongs. The granule will be deleted when its collection is deleted. Much information associated with the granule is defined or restricted by its collection. A granule can be associated with its collection by using the Data Set ID instead of short name and version number.

7.2.7.2.2 Basic Granule Information

The required elements for granule metadata are GranuleUR, InsertTime, LastUpdate and CollectionMetaData where GranuleUR is the granule universal reference id and CollectionMetaData is the associated collection. Additional elements include Data File Size (SizeMBDataGranule), Processing Information (ReprocessingPlanned, ReprocessingActual), Local granule reference (ProducerGranuleID), Day Night Flag (DayNightFlag), PGE information (PGEVersionClass), Temporal Information (RangeDateTime or SingleDateTime), Restriction Settings (RestrictionFlag and RestrictionComment) and can help describe a provider's granule metadata for use in data searches.

The RestrictionFlag indicates whether there is any access constraints applied when the granule data goes public. If the restriction is not explicitly indicated for the granule, the granule inherits the restriction setting of its associated collection.

The InsertTime is the day and time the granule metadata was inserted in the provider's database. The LastUpdate is the day and time the granule metadata was last updated in the provider's database.

Two types of temporal information represent the time when granule data were collected. One type is single day/time and the other type is range day/time. A granule's temporal should be in the range of its collection's temporal and should refer to the same temporal system (UTC/Gregorian for specific). Since the temporal system definition is provided by its primary collection, in granule's metadata input, the temporal system definition is not present.

Temporal information is an essential piece of metadata for a granule search.

Code Example 188. Granule Information

```
<GranuleURMetaData>
   <GranuleUR>SC:MODMGGAD.001:81677</GranuleUR>
   <DbID>81677</DbID>
   <InsertTime>2001-09-20 11:43:31.996</InsertTime>
   <LastUpdate>2001-09-20 11:43:31.996</LastUpdate>
   <CollectionMetaData>
      <ShortName>MODMGGAD</ShortName>
      <VersionID>1</VersionID>
   </CollectionMetaData>
   <DataGranule>
      <SizeMBDataGranule>36.332</SizeMBDataGranule>
      <ReprocessingPlanned>further update is anticipated</ReprocessingPlanned>
      <ReprocessingActual>reprocessed</ReprocessingActual>
      <ProducerGranuleID>MODMGGAD.A2001149.h08v07.003.hdf</ProducerGranuleID>
      <DayNightFlag>Day </DayNightFlag>
      <ProductionDateTime>2001-07-24 12:30:03.0</ProductionDateTime>
      <LocalVersionID>3.0.2</LocalVersionID>
   </DataGranule>
   <PGEVersionClass>
      <PGEVersion>3.0.2 </PGEVersion>
   </PGEVersionClass>
   <RangeDateTime>
      <RangeEndingTime>17:20:00.000000</RangeEndingTime>
      <RangeEndingDate>2001-05-29 00:00:00.0</RangeEndingDate>
      <RangeBeginningTime>17:10:00.000000</RangeBeginningTime>
      <RangeBeginningDate>2001-05-29 00:00:00.0</RangeBeginningDate>
   </RangeDateTime>
</GranuleURMetaData>
```

7.2.7.2.3 Spatial

The spatial data is the spatial area that granule's data covers. This is an essential piece of information for granule search criteria. The spatial coverage area for a granule should be within the spatial coverage area of its primary collection. Detailed handling and expression of spatial information please refer to the Spatial Information section.

7.2.7.2.4 Sources and Sensors

ECHO adopts a layered representation of sources and sensors information for the granule. A source and sensor layer is defined as:

Platform->* Instrument->* Sensor

A granule could be associated with 0 to more then one platforms, each platform could contain 0 to more then one instruments, and each instrument could contain 0 to more then one sensors. There won't be characteristic parameters associated with the platforms and instruments. Granule's platforms' and instruments' characteristic were defined by its primary collection. Granule could define sensor level characteristic value for the characteristic parameters defined by its primary collection. In addition to the characteristic parameters, there might be an operational mode associated with instruments independent of its primary collection. When the data provider does not have the platform concept with their granule, the input for platform short name should be "No Platform Associated". When the data provider does not have the instrument concept for their granule, the input for instrument short name should be "No Instrument Associated". Although in the current version of ECHO granule DTD, there is the entry for platform/instrument characteristics and sensor level characteristics definition, following the rules stated above for granule sources/sensor information integrity is strongly recommended.

Full platform/instrument/sensor description.

Code Example 189. Sources and Sensors

```
<Platform>
   <PlatformShortName>Terra</PlatformShortName>
   <Instrument>
      <InstrumentShortName>MODIS</InstrumentShortName>
      <OperationMode>Operation Mode</OperationMode>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 190. No platform association.

```
<Platform>
   <PlatformShortName>No Platform Associated</PlatformShortName>
   <Instrument>
      <InstrumentShortName>MODIS</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 191. No instrument association.

```
<Platform>
   <PlatformShortName>Terra</PlatformShortName>
   <Instrument>
      <InstrumentShortName>No Instrument Associated</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 192. Sensor only.

```
<Platform>
   <PlatformShortName>No Platform Associated</PlatformShortName>
   <Instrument>
      <InstrumentShortName>No Instrument Associated</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

The platform, instrument, and sensor referred to by the granule is restricted by its primary collection source/sensor reference. The requirement for platform/instrument/sensor entry is compliant with GCMD standard valid. For GCMD standard sources/sensor valid, please view http://gcmd.gsfc.nasa.gov/Resources/valids/index.html

7.2.7.2.5 Additional Attributes

Additional Attributes is a group of parameters that further describes the granule with specific value. These are important search criteria for the granule. The granule's Additional Attributes should be bound with the Additional Attributes defined for its associated collection.

Granules and Additional Attributes.

Code Example 193. Granule-level Additional Attributes

```
<AdditionalAttributes>
          <AdditionalAttribute>
       <AdditionalAttributeName>CalibrationQuality</AdditionalAttributeName>
            <AdditionalAttributeValue>marginal</AdditionalAttributeValue>
          </AdditionalAttribute>
          <AdditionalAttribute>
            <AdditionalAttributeName>MissionPhase</AdditionalAttributeName>
            <AdditionalAttributeValue>A+E</AdditionalAttributeValue>
          </AdditionalAttribute>
          <AdditionalAttribute>
            <AdditionalAttributeName>AveragedBlackBodyTemperature</AdditionalAttributeName>
            <AdditionalAttributeValue>290.0</AdditionalAttributeValue>
          </AdditionalAttribute>
          <AdditionalAttribute>
            <AdditionalAttributeName>NadirPointing</AdditionalAttributeName>
            <AdditionalAttributeValue>Y</AdditionalAttributeValue>
          </AdditionalAttribute>
</ AdditionalAttributes>
```

7.2.7.2.6 Measured Parameters

Measured parameters are associated with a granule only and are important information for the granule. For some providers, the value of certain measured parameters decides the visibility of the granule.

Measured parameters.

Code Example 194. Measured Parameters

```
<GranuleURMetaData>
……
<MeasuredParameter>
   <MeasuredParameterContainer>
      <ParameterName>MODMGGAD</ParameterName>
      <QAStats>
         <QAPercentMissingData>0</QAPercentMissingData>
      </QAStats>
      <QAFlags>
         <AutomaticQualityFlag>Passed</AutomaticQualityFlag>
         <AutomaticQualityFlagExplanation>output file is created and good</AutomaticQualityFlagExplanation>
         <ScienceQualityFlag>Not Investigated</ScienceQualityFlag>
         <ScienceQualityFlagExplanation>See http://modland.nascom.nasa.gov/QA_WWW/release.html for the Science QA status of
this product.</ScienceQualityFlagExplanation>
      </QAFlags>
   </MeasuredParameterContainer>
</MeasuredParameter>
……
</GranuleURMetaData>
```

7.2.7.2.7 Orbital Information

Another piece of interesting information for a granule is the information associated with a satellite's orbit.

Orbit information.

Code Example 195. Orbital Information

```
<GranuleURMetaData>
……
<OrbitCalculatedSpatialDomain>
   <OrbitCalculatedSpatialDomainContainer>
      <OrbitNumber>7694</OrbitNumber>
      <EquatorCrossingLongitude>-100.187</EquatorCrossingLongitude>
      <EquatorCrossingDate>2001-05-29 00:00:00.0</EquatorCrossingDate>
      <EquatorCrossingTime>17:19:38.910554</EquatorCrossingTime>
   </OrbitCalculatedSpatialDomainContainer>
</OrbitCalculatedSpatialDomain>
……
</GranuleURMetaData>
```

7.2.7.2.8 Other Descriptive Information

The other granule metadata includes: Campaign information, Browse, Online URL, and Processing/QA/Input historical information. A granule could associate with 0 to many browse files and a browse file could be referenced by more than one granule.

7.2.7.2.9 Restriction Flag

On a per granule basis, set a value (integer) for the RestrictionFlag.

The RestrictionFlag is used in combination with a provider created data access rule to restrict access to granules based on the value of the restriction flag. If the restriction flag is not set for a granule and there is no collection default then the data access Rule will not be applied to the granule.

Providers are completely free to use any set of numbers (as long as they are integers) and assign whatever meaning they like to the numbers.

An example might be to use a RestrictionFlag for a data quality summary, with a range of values 0-10, 0 being unknown, 1 being bad and 10 excellent. Combine with a data access rule that restricts access to granules with a RestrictionFlag less than or equal to 5. Guest users will only be allowed to view granules with a data quality summary of 6 or better.

Setting a restriction flag.

Code Example 196. Restriction Flag set for a Granule

```
            <RestrictionFlag>8</RestrictionFlag>
      <RestrictionComment>quality summary value of 8 (good)</RestrictionComment>
```

Note: Restriction flag set for a granule overrides the default Collection value.

7.2.7.2.10 Two Dimensional Coordinate System

Granule specific values will be searched against by a two-dimensional coordinate system-based catalog search for Coordinate1 and Coordinate2. Examples of two-dimensional coordinate system values are path/row for WRS data and MODIS tiles ids.

Note: Two-dimensional coordinate system types are defined by ECHO operations and have to be enabled for a collection before it can be used in granule ingest.

Code Example 197. Setting Two-dimensional coordinate system coordinates

```
            <TwoDCoordinateSystem>
        <StartCoordinate1>21</StartCoordinate1>
        <StartCoordinate2>29</StartCoordinate2>
        <EndCoordinate2>33</EndCoordinate2>
        <TwoDCoordinateSystemName>WRS2</TwoDCoordinateSystemName>
      </TwoDCoordinateSystem>
```

7.2.8 *Spatial Representations, Coordinates & Projections*

For a collection or granule, the spatial area coverage is one of the most important and basic search criteria for earth science data, although this is not required data. There are some specific issues that need to be discussed to enable the user to submit the metadata correctly.

ECHO, currently, supports multiple spatial representations. Each provider is allowed to have different spatial representation; however, each collection must define a single spatial representation for its granules. The following sections cover details on the spatial representations.

ECHO currently supports two coordinate systems for spatial data. Each data provider should use only one coordinate system when constructing the spatial area coverage for a collection or granule.

7.2.8.1 Geometry Representations

Spatial data are most commonly described as geometry such as a polygon, a multi-polygon, or a line. They are stored as Oracle spatial objects in the database to record shape, spatial locations of corner points and spatial coordinate system used, Cartesian or Geodetic. Cartesian and Geodetic are considered different spatial representations. Corner points in the Cartesian system are connected by straight lines, while in the Geodetic system they are connected by great circle arcs. This makes their spatial coverages slightly different. Oracle requires specifying the coordinate system during data storage time. Therefore, data under different coordinate systems are also searched differently.

7.2.8.2 Coordinate System

7.2.8.2.1 Cartesian Coordinate System

The Cartesian Coordinate System is a flattened coordinate system with longitude ranged from –180 to 180 degrees and latitude ranged from –90 to 90 degrees. The projected map is flattened and open along the International Date Line with North Pole and South Pole as top and bottom line respectively.

7.2.8.2.2 Geodetic Coordinate System

The Geodetic coordinate system is defined in angular (latitude and longitude) and is defined relative to spherical polar coordinate and Earth Geodetic datum. Oracle defines coordinate system following OGC standards. The Geodetic coordinate ECHO chose to support is WGS 84. It is defined as:

GEOGCS ["Longitude / Latitude (WGS 84)", DATUM ["WGS 84", SPHEROID ["WGS 84", 6378137.000000, 298.257224]], PRIMEM ["Greenwich", 0.000000], UNIT ["Decimal Degree", 0.01745329251994330]]

7.2.8.3 Data Types and Representation

7.2.8.3.1 Point

ECHO can receive, store, and support the search on spatial data representing one or more points. To send ECHO spatial point data in the metadata XML file, the information is expressed as follows.

Single Point example.

Code Example 198. Single Point.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Point>
         <PointLongitude>-123.948</PointLongitude>
         <PointLatitude>45.0664</PointLatitude>
      </Point>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

Code Example 199. Multiple points.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Point>
         <PointLongitude>-123.948</PointLongitude>
          <PointLatitude>45.0664</PointLatitude>
      </Point>
      <Point>
         <PointLongitude>-133.546</PointLongitude>
         <PointLatitude>45.0664</PointLatitude>
      </Point>
   </HorizontalSpatialDomainContainer>
```

```
</SpatialDomainContainer>
```

7.2.8.3.2 Line

ECHO can receive, store, and search spatial data representing one or more lines. To send ECHO spatial line data in the metadata XML file, the information is expressed as:

Single line example.

Code Example 200. Single line.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Line>
         <Point>
            <PointLongitude>-123.948</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>-133.546</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
      </Line>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer >
```

Code Example 201. Multiple lines.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Line>
         <Point>
            <PointLongitude>-123.948</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>-133.546</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
      </Line>
      <Line>
         <Point>
            <PointLongitude>-123.948</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>-133.546</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>-143.546</PointLongitude>
            <PointLatitude>40.0664</PointLatitude>
         </Point>
      </Line>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer >
```

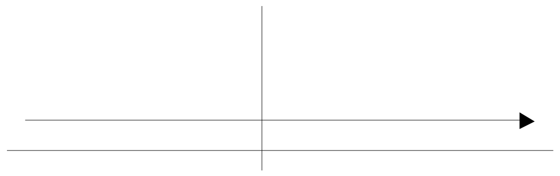
In the Cartesian coordinate system, the points are connected with a straight line on the plane in the order listed. The line connects two points that will never cross the International Date Line or Poles.

Code to be interpreted in Cartesian and Geodetic systems.

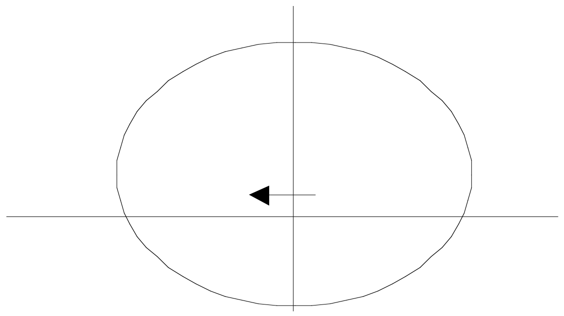
Code Example 202.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Line>
         <Point>
            <PointLongitude>-173.948</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>173.546</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
      </Line>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

The figure below shows a line in the Cartesian coordinate system:



This same expression in the Geodetic coordinate system represents the line as:



In the Geodetic coordinate system, the line connects two points using a great circle arc with shortest distance between the two points. The line could cross the International Date Line and Poles. If the same line indicated in the Cartesian coordinate system is represented in the Geodetic coordinate system, the expression should be

Example of Adding density.

Code Example 203. Adding density

```
<SpatialDomainContainer>
```

```
   <HorizontalSpatialDomainContainer>
      <Line>
         <Point>
            <PointLongitude>-173.948</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>0.0</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>173.546</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
      </Line>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer >
```

The additional point gives the line more density so that Oracle interpolates the data correctly. The appropriate density should be applied to other spatial type for Geodetic coordinate system as well.

7.2.8.3.3 Polygon

ECHO is capable of receiving, storing, and searching spatial data representing a polygon, a polygon with hole, and multiple polygons (any type – with hole or without hole). To send ECHO spatial polygon data in the metadata XML file, the information is expressed as:

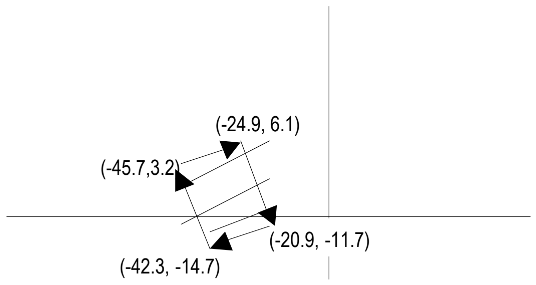
Single polygon example.

Code Example 204. Single polygon

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-20.9342</PointLongitude>
                     <PointLatitude>-11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-42.3067</PointLongitude>
                     <PointLatitude>-14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-45.7985</PointLongitude>
                     <PointLatitude>3.198</PointLatitude>
                  </Point>
                  <Point>
                    <PointLongitude>-24.8982</PointLongitude>
                    <PointLatitude>6.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
            </SinglePolygon>
        </Polygon>
    </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

A single polygon can only have one outer ring that represents the area surrounded within. The points are connected using a straight line on a Cartesian coordinate system in the order listed. In the Cartesian coordinate system, the points should be listed in the order of clockwise connection. The area should not cross the International Date Line or Poles. In the Geodetic coordinate system, the points are connected using a great circle arc according to the shortest distance between the two points. The density concept is especially important for polygon representation in the Geodetic coordinate system. The polygon coverage could cross the International Date Line and/or poles in the Geodetic coordinate system.

The spatial area covered by the above representation is:



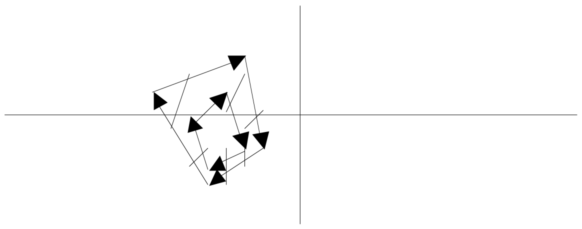
Single polygon with hole.

Code Example 205. Single polygon with hole

```
<SpatialDomainContainer>
    <HorizontalSpatialDomainContainer>
        <Polygon>
            <SinglePolygon>
                <OutRing>
                    <Boundary>
                        <Point>
                            <PointLongitude>-20.9342</PointLongitude>
                            <PointLatitude>-11.7045</PointLatitude>
                        </Point>
                        <Point>
                            <PointLongitude>-42.3067</PointLongitude>
                            <PointLatitude>-14.7732</PointLatitude>
                        </Point>
                        <Point>
                            <PointLongitude>-45.7985</PointLongitude>
                            <PointLatitude>3.198</PointLatitude>
                        </Point>
                        <Point>
                            <PointLongitude>-24.8982</PointLongitude>
                            <PointLatitude>6.1665</PointLatitude>
                        </Point>
                    </Boundary>
                </OutRing>
                <InnerRing>
                    <Boundary>
                        <Point>
                            <PointLongitude>-22.9342</PointLongitude>
                            <PointLatitude>-5.7045</PointLatitude>
                        </Point>
                        <Point>
                            <PointLongitude>-30.3067</PointLongitude>
                            <PointLatitude>-10.7732</PointLatitude>
                        </Point>
                        <Point>
                            <PointLongitude>-35.7985</PointLongitude>
                            <PointLatitude>1.198</PointLatitude>
                        </Point>
                        <Point>
                            <PointLongitude>-30.8982</PointLongitude>
                            <PointLatitude>3.1665</PointLatitude>
                        </Point>
                    </Boundary>
                </InnerRing>
            </SinglePolygon>
        </Polygon>
    </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

A single polygon with a hole can only have one outer ring that represents the area surrounded within. A single polygon with a hole can have multiple inner rings that represent holes. All the rules, restrictions, and discussions for the outer ring in both coordinate systems apply to inner ring. An inner ring should be completely contained by an outer ring.

The spatial area covered by above representation is:



Multiple Polygons:

A multiple polygon is the combination of polygons. Those single polygons could be with or without holes. Each single polygon expression should follow the same rules listed above. No two polygons may overlap each other.

7.2.8.3.4 Bounding Box

ECHO is capable of receiving, storing, and supporting the search on spatial data representing a bounding box or multiple bounding boxes. To send ECHO spatial point data in the metadata XML file, the information is expressed as:

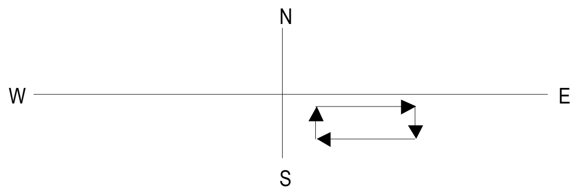
Example of using a bounding box.

Code Example 206. Bounding box

```
<HorizontalSpatialDomainContainer>
    <BoundingRectangle>
        <WestBoundingCoordinate>8.733</WestBoundingCoordinate>
        <NorthBoundingCoordinate>-7.4861</NorthBoundingCoordinate>
```

```
      <EastBoundingCoordinate>43.199501</EastBoundingCoordinate>
      <SouthBoundingCoordinate>-35.2617</SouthBoundingCoordinate>
   </BoundingRectangle>
</HorizontalSpatialDomainContainer>
<HorizontalSpatialDomain>
   <ZoneIdentifier/>
     <BoundingRectangle>
        <WestBoundingCoordinate><8.733></WestBoundingCoordinate>
        <NorthBoundingCoordinate><-7.4861></NorthBoundingCoordinate>
        <EastBoundingCoordinate><43.199501></EastBoundingCoordinate>
       <SouthBoundingCoordinate><-35.2617></SouthBoundingCoordinate>
     </BoundingRectangle>
</HorizontalSpatialDomain>
```

The spatial area coverage is:



ECHO stores a bounding box as a four pointed polygon, subject to the specifications and constraints described in the Polygon section. In the Cartesian system, bounding boxes cannot cross the International Date Line or poles. In the Geodetic coordinate system, bounding box coverage is not allowed to have an area greater than or equal to half the area of the Earth. If there is more than one bounding box listed, then ECHO stores them as multiple polygons which cannot overlap.

Note: Data providers that use the Geodetic coordinate system for their spatial data should be particularly cognizant of the Geodetic "half Earth" restriction described above. These data providers should use polygons rather than bounding boxes when describing their data.

7.2.8.4 Invalid Spatial Representation

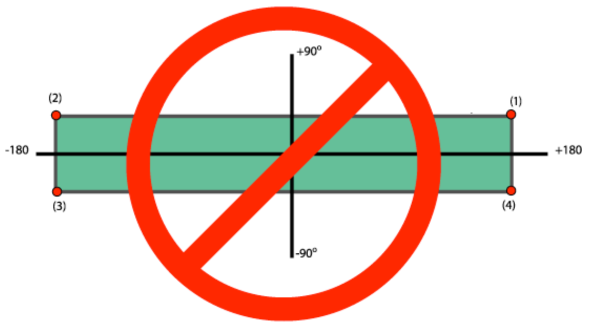
7.2.8.4.1 Polygon with points in counter-clockwise order

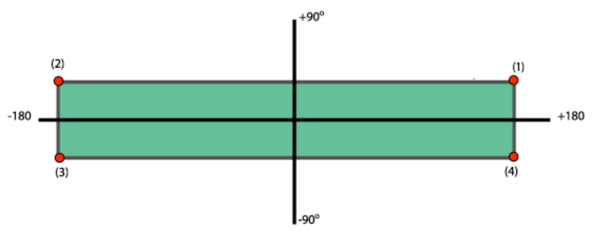
7.2.8.4.2

Code Example 207. Polygon with points in counter-clockwise order.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>170</PointLongitude>
                     <PointLatitude>30</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-170</PointLongitude>
                     <PointLatitude>30</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-170</PointLongitude>
                     <PointLatitude>-30</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>170</PointLongitude>
                     <PointLatitude>-30</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

This spatial area expression is considered invalid in the Cartesian coordinate system because of the order of the points shown on the plane. However, this same expression is considered valid in the Geodetic coordinate system, although the coverage will be interpreted differently.





Cartesian Geodetic

This expression is invalid in the Cartesian coordinate system,

but valid in the Geodetic coordinate system.

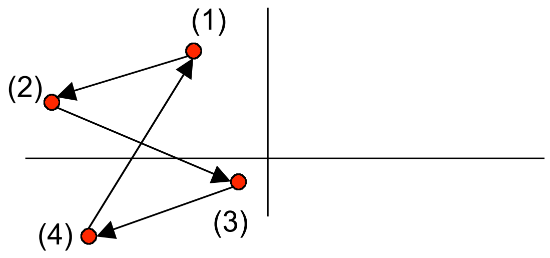
7.2.8.4.3 Twisted Polygon

Code Example 208. Twisted polygon.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-20.9342</PointLongitude>
                     <PointLatitude>-11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-42.3067</PointLongitude>
                     <PointLatitude>-14.7732</PointLatitude>
                  </Point>
                  <Point>
```

```
                    <PointLongitude>-24.8982</PointLongitude>
                    <PointLatitude>6.1665</PointLatitude>
                </Point>
                <Point>
                    <PointLongitude>-45.7985</PointLongitude>
                    <PointLatitude>3.198</PointLatitude>
                </Point>
            </Boundary>
        </OutRing>
    </SinglePolygon>
  </Polygon>
 </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

This presentation shows a polygon as:



Twisted polygon.

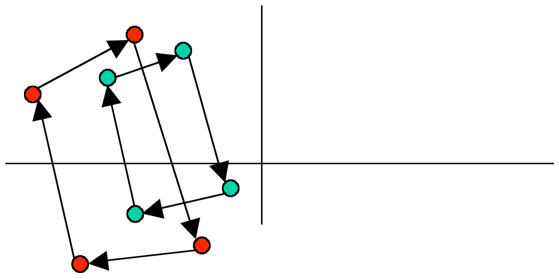
In either coordinate system, this expression represents an invalid polygon.

7.2.8.4.4 Hole crosses over outer ring

Code Example 209. Hole crosses the outer ring.

```
<SpatialDomainContainer>
  <HorizontalSpatialDomainContainer>
    <Polygon>
      <SinglePolygon>
        <OutRing>
          <Boundary>
            <Point>
              <PointLongitude>-20.9342</PointLongitude>
              <PointLatitude>-11.7045</PointLatitude>
            </Point>
            <Point>
              <PointLongitude>-42.3067</PointLongitude>
              <PointLatitude>-14.7732</PointLatitude>
            </Point>
            <Point>
              <PointLongitude>-45.7985</PointLongitude>
              <PointLatitude>3.198</PointLatitude>
            </Point>
            <Point>
              <PointLongitude>-24.8982</PointLongitude>
              <PointLatitude>6.1665</PointLatitude>
            </Point>
```

```
                </Boundary>
            </OutRing>
            <InnerRing>
                <Boundary>
                    <Point>
                        <PointLongitude>-17.9342</PointLongitude>
                        <PointLatitude>-5.7045</PointLatitude>
                    </Point>
                    <Point>
                        <PointLongitude>-30.3067</PointLongitude>
                        <PointLatitude>-10.7732</PointLatitude>
                    </Point>
                    <Point>
                        <PointLongitude>-35.7985</PointLongitude>
                        <PointLatitude>1.198</PointLatitude>
                    </Point>
                    <Point>
                        <PointLongitude>-10.8982</PointLongitude>
                        <PointLatitude>3.1665</PointLatitude>
                    </Point>
                </Boundary>
            </InnerRing>
        </SinglePolygon>
    </Polygon>
  </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```



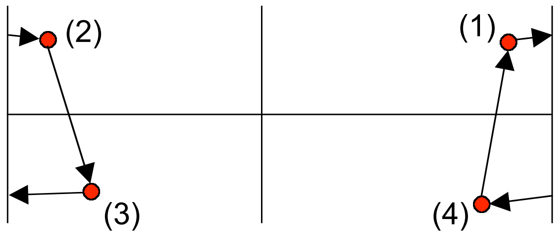
Hole crosses outer ring.

In either coordinate system, this represents an invalid spatial area.

7.2.8.4.5 Polygon Crosses International Date Line

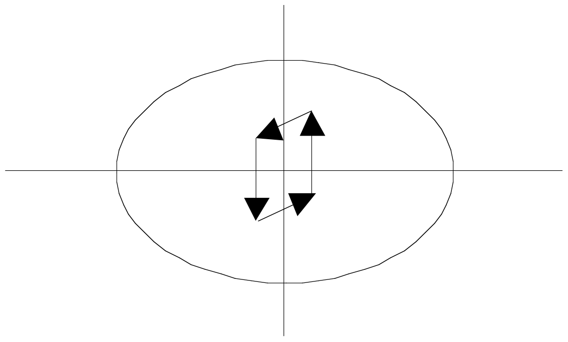
Code Example 210. Polygon crosses the International Date Line.

```
<SpatialDomainContainer>
  <HorizontalSpatialDomainContainer>
    <Polygon>
        <SinglePolygon>
            <OutRing>
                <Boundary>
                    <Point>
                        <PointLongitude>170.9342</PointLongitude>
                        <PointLatitude>11.7045</PointLatitude>
                    </Point>
                    <Point>
                        <PointLongitude>-175.3067</PointLongitude>
                        <PointLatitude>14.7732</PointLatitude>
                    </Point>
                    <Point>
                        <PointLongitude>-176.7985</PointLongitude>
                        <PointLatitude>-13.198</PointLatitude>
                    </Point>
                    <Point>
                        <PointLongitude>172.8982</PointLongitude>
                        <PointLatitude>-7.1665</PointLatitude>
                    </Point>
                </Boundary>
            </OutRing>
        </SinglePolygon>
    </Polygon>
  </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```



Polygon crosses International Date Line.

According to clockwise order, this expression in the Cartesian coordinate system represents a polygon crossing the International Date Line. Thus, it is invalid. However, in the Geodetic coordinate system, this is a valid spatial coverage area. The coverage area is as shown:

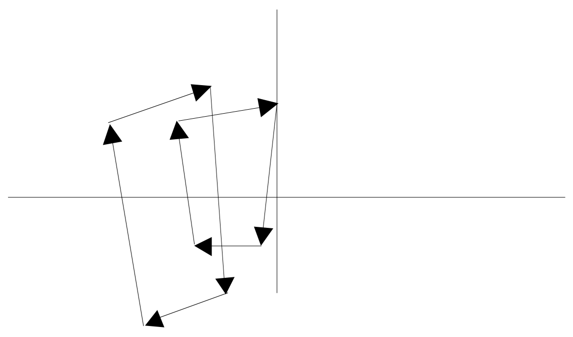


7.2.8.4.6 Overlapping Polygons

Code Example 211. Overlapping polygons.

```
<SpatialDomainContainer>
  <HorizontalSpatialDomainContainer>
    <Polygon>
      <SinglePolygon>
```

```
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-20.9342</PointLongitude>
                     <PointLatitude>-11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-42.3067</PointLongitude>
                     <PointLatitude>-14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-45.7985</PointLongitude>
                     <PointLatitude>3.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-24.8982</PointLongitude>
                     <PointLatitude>6.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-2.9342</PointLongitude>
                     <PointLatitude>-5.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-25.3067</PointLongitude>
                     <PointLatitude>-5.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-25.7985</PointLongitude>
                     <PointLatitude>3.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-0.8982</PointLongitude>
                     <PointLatitude>4.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

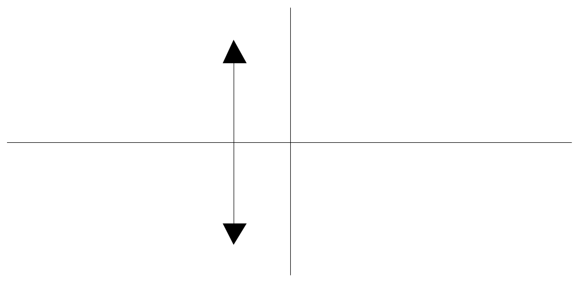


In either coordinate system, this expression represents an invalid spatial data.

7.2.8.4.7 Inappropriate Data

Code Example 212. Inappropriate data.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <BoundingRectangle>
         <WestBoundingCoordinate>-8.733</WestBoundingCoordinate>
         <NorthBoundingCoordinate>7.4861</NorthBoundingCoordinate>
         <EastBoundingCoordinate>-8.733</EastBoundingCoordinate>
         <SouthBoundingCoordinate>10.2617</SouthBoundingCoordinate>
      </BoundingRectangle>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

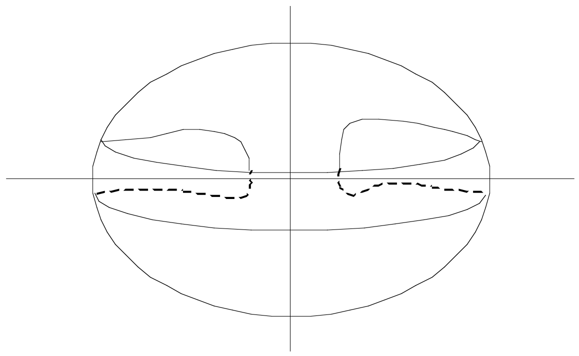


In this example, the bounding box is actually a line. It is important to use the correct spatial data type when constructing a spatial data expression. Do not use a bounding box to express a line.

Note: The granularity of the spatial index for a given provider can result in two distinct but nearby points being interpreted as the same point.

7.2.8.4.8 Incorrect Density

If a spatial coverage area as shown below is intended in a Geodetic coordinate system:



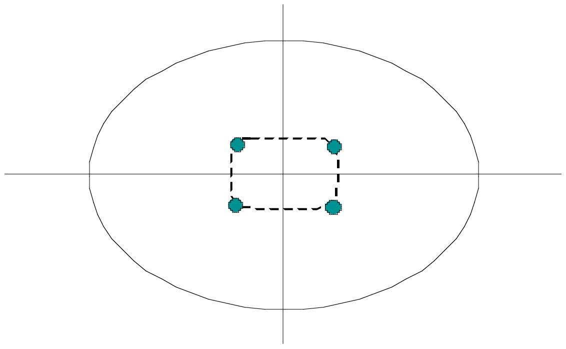
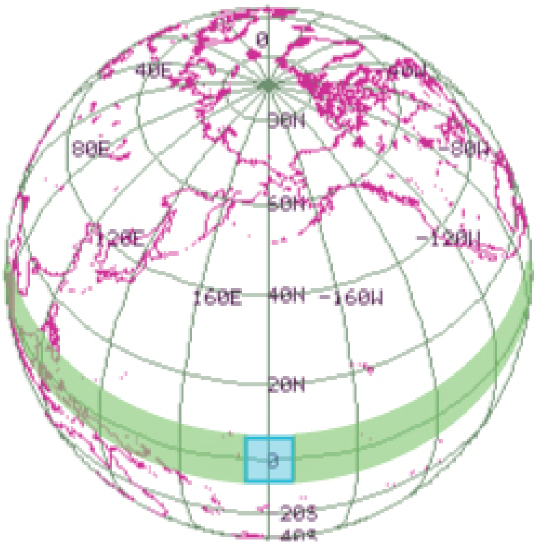
The expression in the input file is shown as below:

Example of incorrect density.

Code Example 213. Incorrect density

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>170.9342</PointLongitude>
                     <PointLatitude>11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-175.3067</PointLongitude>
                     <PointLatitude>14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-176.7985</PointLongitude>
                     <PointLatitude>-13.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>172.8982</PointLongitude>
                     <PointLatitude>-7.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

This expression in a Geodetic coordinate system is a valid spatial data. However, the spatial coverage area represented will be as shown below:

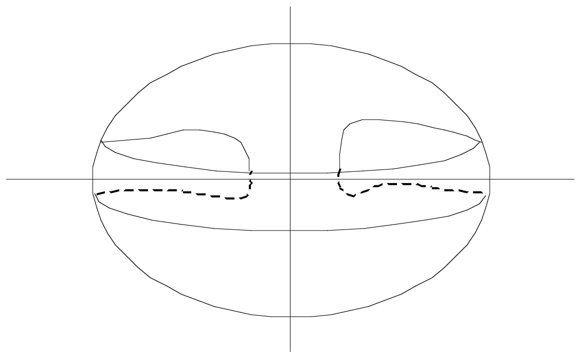
You may want the large green band, but you will get the small blue rectangle.

Oracle connects any two points using shortest distance between the points. To correctly represent this spatial coverage, additional points are required to increase the density so that the spatial area can be represented. One of the expressions with additional points that will represent this spatial coverage area correctly is as shown below:

Example of using the correct density.

Code Example 214.   Correct density

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>170.9342</PointLongitude>
                     <PointLatitude>11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>0.0</PointLongitude>
                     <PointLatitude>14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-175.3067</PointLongitude>
                     <PointLatitude>14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-176.7985</PointLongitude>
                     <PointLatitude>-13.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>0.0</PointLongitude>
                     <PointLatitude>-10.1665</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>172.8982</PointLongitude>
                     <PointLatitude>-7.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```



7.2.8.4.9 Polygon Coverage Larger then Half of the Earth

7.2.8.4.10

Code Example 215. Polygon covers more than half the Earth.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>170.9342</PointLongitude>
                     <PointLatitude>71.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>0.0</PointLongitude>
                     <PointLatitude>71.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-175.3067</PointLongitude>
                     <PointLatitude>71.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-176.7985</PointLongitude>
                     <PointLatitude>-71.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>0.0</PointLongitude>
                     <PointLatitude>-71.1665</PointLatitude>
                  </Point>
```

```
                <Point>
                    <PointLongitude>172.8982</PointLongitude>
                    <PointLatitude>-71.1665</PointLatitude>
                </Point>
            </Boundary>
        </OutRing>
      </SinglePolygon>
    </Polygon>
  </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

The spatial area coverage in this case is much bigger then half of the earth, thus this is not a valid spatial data in a Geodetic coordinate system. This is not a valid spatial data in a Cartesian coordinate system either because the points are not expressed in clockwise order. If the points were listed in reverse order, this spatial data would be a valid record in Cartesian coordinate system.

7.2.8.5 Latitude/Longitude Range and Tolerance

Tolerance and range parameter settings are used to associate a level of precision and data range with spatial data. These parameters are used as part of the evaluation parameter when validating spatial data input. For any spatial data expressed using latitude/longitude, the range is expressed in degrees and the tolerance in meters. For example, if the range for latitude is –50 to 50 and the range for longitude is –150 to 150, then any input data with latitude and/or longitude outside of this range is considered invalid. If the tolerance is 0.05 for both latitude and longitude, and if the distance between two points is less than 0.05 meter both longitudinally and latitudinally, then those two points are considered the same point. In this situation, the spatial expression is invalid because ECHO spatial constructs require each point to have unique spatial locations.

Range and tolerance parameters should be defined specifically for all data. ECHO defaults for range and tolerance are:

Latitude: -90 to 90

Longitude: -180 to 180

Tolerance for both latitude and longitude: 0.05 meters.

7.2.9 Online Data Access URL and Online Resources URL

There could be much online information available for a collection or granule, such as guides, product listings, validation information, etc. For certain granules or collections, the raw data are made available online, via FTP or web URL. The URL information associated with these collections or granules could be sent to ECHO and made available to the user. ECHO chooses to store this online access information for directly accessible granule and collection data differently from information covering other aspects of granule and collection data. Directly accessible data will require that a data provider use the <OnlineAccessURLs> tag and include the URL to that data. Any other information covering aspects of that data will require the use of the <CollectionOnlineResources> or <GranuleOnlineResources> tags, along with the URLs to that information.

7.2.10 Ingest Reporting

The automated ingest process will generate an ingest summary report for each provider with ingest start/stop time information, data files received with the file size respectively, number of collection/granule processed, etc. The ingest report will be sent to designated provider personnel (specified as the ProviderContact when the provider registered with ECHO) via e-mail in XML format.

7.2.11 Packaging/Shipping Options

A new provider needs to specify what ordering options are made available to clients when orders are placed. Providers should contact the ECHO Operations team to have order options configured in the ECHO system. Refer to the Orders and Options section for a discussion of possible options.

7.2.12 New Items vs. Update Items

ECHO expects a complete set of granule or collection metadata for both inserts and updates. When processing a granule or collection, the ECHO system first checks to see if the item already exists. If the item already exists in the system, all the data associated with this item will be deleted before the new data is inserted into the ECHO database. The item identification is used to search for existing data. For a granule, the GranuleUR is assumed to be a unique identifier for the data provider, while for collections the short name plus version number are assumed to be a unique identifier for the data provider. The ECHO system applies the same principle to deal with the insertion or update of the granule or collection when processed against the XML metadata input. When updating a collection or a granule, only the one with most recent update time will be stored in the ECHO database. Any collection/granule received from the provider with a last update date earlier then the records in the ECHO database is ignored.

7.2.13 Delete Items

Data providers will instruct ECHO to delete collections or granules by placing the collections or granules under the <DeleteCollections> or <DeleteGranules> tag. Only the item's identification is used to process the deletion of the item and all the data associated with this item. The deleted items' identification and deletion date will be kept in the ECHO database for data history auditing purpose. The only way to re-install the collections or granules in the ECHO system is to re-submit the metadata for those items to ECHO for insertion.

7.2.14 Data Not Included in DTD

The DTD does not provide descriptions of certain data characteristics such as data type, length of field, etc. This additional information is available in the ECHO data dictionary located on the ECHO Web site at http://eos.nasa.gov/echo. If data received does not meet the specifications described in the data dictionary, those data records will be ignored and could cause ingest of the file containing these data records to fail.

For any repetitive item identification in the metadata input, only one item bearing the most recent update date with its complete information will be ingested into the database. The rest of the items and their associated information will be ignored even if they contain different data information and will result in a duplicate error in the ingest summary for that item.

7.2.15 Metadata Update DTD

To update data, the provider has to follow ECHO's metadata update DTD and some other rules for the metadata update XML files.

Currently, ECHO handles a) granule level Online Access URL (delete, update, insert) and Measured Parameter QA Flags (update) metadata update and b) browse update (see section on browse below). Collection level metadata updates are ignored.

Code Example 180 Metadata update DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT ProviderAccountService (UpdateMetadata)>
<!--UpdateMetadata can update a single collection, multiple collections, a single granule, or multiple granules in one
transaction.  Each update allows the addition of new metadata-->
<!ELEMENT UpdateMetadata (Collection*, Granule*)>
<!ELEMENT Collection (Target+, (Add | Update | Delete)+)>
<!ELEMENT Granule (Target+, (Add | Update | Delete)+)>
<!-- Target+ allows the same change to be made to several different granules or collections simultaneously.  This is
especially useful for bulk deletions of OnlineURLs. -->
<!ELEMENT Target (ID, ProviderLastUpdateDateTime, SaveDateTimeFlag?)>
<!-- SaveDateTimeFlag is the flag that allows echo to update the last update date time for Target. The default is SAVE -->
<!ELEMENT Add (QualifiedTag, MetadataValue)>
<!ELEMENT Update (QualifiedTag, MetadataValue)>
<!ELEMENT Delete (QualifiedTag+)>
<!ELEMENT QualifiedTag (#PCDATA)>
<!ELEMENT MetadataValue (#PCDATA)>
```

```
<!ELEMENT ProviderLastUpdateDateTime (#PCDATA)>
<!ELEMENT SaveDateTimeFlag (SAVE | DONTSAVE)>
<!ELEMENT SAVE EMPTY>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT DONTSAVE EMPTY>
```

AdditionalAttributes element in Metadata Model

```
<!ELEMENT AdditionalAttributes (AdditionalAttribute)+>
<!ELEMENT AdditionalAttribute (AdditionalAttributeName, AdditionalAttributeValue*, AdditionalAttributeValueType?)>
<!ELEMENT AdditionalAttributeName (#PCDATA)>
<!ELEMENT AdditionalAttributeValue (#PCDATA)>
<!ELEMENT AdditionalAttributeValueType (#PCDATA)>
```

The QualifiedTag is a string that follows the XPath standard. It is used to indicate which item in the DTD representing ECHO's metadata model is to be manipulated (e.g., updated, inserted, or deleted). In the case of deleting all online URLs for granules or collections, the XPath need only look like “OnlineAccessURLs”. An optional flag, SaveDateTimeFlag, allows the date/time timestamp that identifies when metadata was last updated to be changed to reflect new update date/time information.

7.2.16 Values for the Qualified Tag for Updates, Deletes, and Inserts

The following DTD excerpts list the acceptable tags used to indicate which field in the ECHO data model is to be updated.

Note: Not all fields in the data model are available for partial update as of ECHO Release 6.0.

Code Example 216. DTD excerpt for collection metadata.

```
<!ELEMENT OnlineAccessURLs (OnlineAccessURL +)>
<!ELEMENT OnlineAccessURL (URL, URLDescription?, MimeType)>
<!ELEMENT CollectionOnlineResources (OnlineResource +)>
<!ELEMENT OnlineResource (OnlineResourceURL, OnlineResourceDescription?, OnlineResourceType, OnlineResourceMimeType)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT URLDescription (#PCDATA)>
<!ELEMENT MimeType (#PCDATA)>
<!ELEMENT OnlineResourceURL(#PCDATA)>
<!ELEMENT OnlineResourceDescription(#PCDATA)>
<!ELEMENT OnlineResourceType (#PCDATA)>
<!ELEMENT OnlineResourceMimeType (#PCDATA)>
```

Code Example 217. DTD excerpt for granule metadata.

```
<!ELEMENT OnlineAccessURLs (OnlineAccessURL +)>
<!ELEMENT OnlineAccessURL (URL, URLDescription?, MimeType)>
<!ELEMENT GranuleOnlineResources (OnlineResource +)>
<!ELEMENT OnlineResource (OnlineResourceURL, OnlineResourceDescription?, OnlineResourceType, OnlineResourceMimeType)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT URLDescription (#PCDATA)>
<!ELEMENT MimeType (#PCDATA)>
<!ELEMENT OnlineResourceURL(#PCDATA)>
```

```
<!ELEMENT OnlineResourceDescription(#PCDATA)>
<!ELEMENT OnlineResourceType (#PCDATA)>
<!ELEMENT OnlineResourceMimeType (#PCDATA)>
```

OnlineAccessURLs are to be used only for the actual data. URLs to all other kinds of Web pages (including metadata) must be sent as OnlineResources.

This table identifies valid XPath values for the qualified tags.

Table 18: Table 17 Valid XPath values for qualified tags.

Tag	XPath	Value
Granule URL Delete One	OnlineAccessURLs/OnlineAccessURL[URL="old_url"]/URL	
Granule URL Delete All	OnlineAccessURLs	
Granule URL Update	OnlineAccessURLs /OnlineAccessURL[URL="old_url"]/URL	New url
Granule URL Insert	OnlineAccessURLs /OnlineAccessURL/URL	url
Granule Online resource Delete One	GranuleOnlineResources/OnlineResource[OnlineResourceURL="old_url"]/ OnlineResourceURL	
Granule Online resource URL Update	GranuleOnlineResources/OnlineResource[OnlineResourceURL="old_url"] / OnlineResourceURL	New url
Granule Online resource type Update	GranuleOnlineResources/OnlineResource[OnlineResourceURL ="old_url"]/OnlineResourceType	New type
Granule Online resource mime type Insert	GranuleOnlineResources/OnlineResource[OnlineResourceURL ="old_url"]/OnlineResourceMimeType	Mime type
Granule Automatic QA Flag Update	MeasuredParameter/MeasuredParameterContainer [ParameterName="p1"]/QAFlags/AutomaticQualityFlag	New flag
Granule Automatic	MeasuredParameter/MeasuredParameterContainer	New explanation

Tag	XPath	Value
QA Flag explanation Update	[ParameterName="p1"]/QAFlags/AutomaticQualityFlagExplanation	
Granule Operational QA Flag Update	MeasuredParameter/MeasuredParameterContainer [ParameterName="p1"]/QAFlags/OperationalQualityFlag	New flag
Granule Operational QA Flag explanation Update	MeasuredParameter/MeasuredParameterContainer [ParameterName="p1"]/QAFlags/OperationalQualityFlagExplanation	New explanation
Granule Science QA Flag Update	MeasuredParameter/MeasuredParameterContainer [ParameterName="p1"]/QAFlags/ScienceQualityFlag	New flag
Granule Science QA Flag explanation Update	MeasuredParameter/MeasuredParameterContainer [ParameterName="p1"]/QAFlags/ScienceQualityFlagExplanation	New explanation
Delete One	AdditionalAttributes/AdditionalAttribute[AdditionalAttributeName="old_name"]/AdditionalAttributeName	
Delete All	AdditionalAttributes	
Update	AdditionalAttributes/AdditionalAttribute[AdditionalAttributeName= "old_name"]/AdditionalAttributeName AdditionalAttributes/AdditionalAttribute[AdditionalAttributeName= "old_name"]/AdditionalAttributeValue AdditionalAttributes/AdditionalAttribute[AdditionalAttributeName= "old_name"]/AdditionalAttributeValueType	new name new value new type
Insert	AdditionalAttributes/AdditionalAttribute/AdditionalAttributeName AdditionalAttributes/AdditionalAttribute[AdditionalAttributeName= "name"]/AdditionalAttributeValue AdditionalAttributes/AdditionalAttribute[AdditionalAttributeName=	Name Value Type

Tag	XPath	Value
	”name”]/AdditionalAttributeValueType	

7.2.17 Browse Image Files and Browse Metadata

Both browse image files and the browse metadata XML file will be sent to the ECHO system. The ECHO system will allocate the storage for browse files, build a browse image URL, and update the database to associate the browse URL to its item record.

When ECHO processes the browse ingest, all the browse image files indicated by the tag <InternalFileName> under the tag of <BrowseCrossReference> in the browse metadata input XML file must be sent as well. In addition to checking the existence of the browse image files, ECHO also verifies the actual browse image file size against the file size indicated for that file in the browse metadata input XML file. If any browse image file indicated in the browse metadata input XML file does not exist, or if any file size does not match the indicated size, then the browse image files and browse metadata input XML file will not be processed.

After processing, ECHO places the browse image files online and associates them with the appropriate granule(s). This information is made available to the end user. The file name in the <InternalFileName> tag is the provider’s unique identifier of the browse file. ECHO supports many-to-many referencing between granules and browse files.

Code Example 218. Browse metadata input XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<BrowseReferenceFile>
  <DTDVersion>1.0</DTDVersion>
  <DataCenterId>EDC</DataCenterId>
  <TemporalCoverage>
    <StartDate>2002-10-17T00:00:00.000Z</StartDate>
    <EndDate>2002-10-18T00:00:00.000Z</EndDate>
  </TemporalCoverage>

  <BrowseCrossReference>
    <GranuleUR>SC:L70RWRS.002:2008440693</GranuleUR>
    <BrowseGranuleId></BrowseGranuleId>
    <InsertTime></InsertTime>
    <LastUpdate>2002-10-17 18:55:33.996</LastUpdate>
    <InternalFileName>:BR:Browse.001:2008440612:1.BINARY</InternalFileName>
    <BrowseSize>167554.0</BrowseSize>
  </BrowseCrossReference>

  <BrowseCrossReference>
    <GranuleUR>SC:L70RWRS.002:2008440702</GranuleUR>
    <BrowseGranuleId></BrowseGranuleId>
    <InsertTime></InsertTime>
    <LastUpdate>2002-10-17 18:56:04.216</LastUpdate>
    <InternalFileName>:BR:Browse.001:2008440624:1.BINARY</InternalFileName>
    <BrowseSize>159823.0</BrowseSize>
  </BrowseCrossReference>

  <BrowseCrossReference>
    <GranuleUR>SC:L70RWRS.002:2008440732</GranuleUR>
    <BrowseGranuleId></BrowseGranuleId>
    <InsertTime></InsertTime>
    <LastUpdate>2002-10-17 18:56:14.856</LastUpdate>
    <InternalFileName>:BR:Browse.001:2008440630:1.BINARY</InternalFileName>
    <BrowseSize>65258.0</BrowseSize>
  </BrowseCrossReference>
</BrowseReferenceFile>
```

7.2.18 Browse Insert, Update, Replace and Delete

This feature extends the current ECHO browse insert capability, and enables the update/replacement and deletion of browse references.

Collection and granule metadata associated with to-be-ingested browse should be in the ECHO database.   Browse data for ingest must conform to the browse dtd described in Section 9.2.

7.2.18.1 Browse DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--DTD generated by XMLSpy v2005 rel. 3 U (http://www.altova.com)-->
<!ELEMENT BrowseReferenceFile (DTDVersion, DataCenterId, TemporalCoverage, DeleteBrowse*, BrowseCrossReference*)>
<!ELEMENT DTDVersion (#PCDATA)>
<!ELEMENT DataCenterId (#PCDATA)>
<!ELEMENT TemporalCoverage (StartDate, EndDate)>
<!ELEMENT StartDate (#PCDATA)>
<!ELEMENT EndDate (#PCDATA)>
<!ELEMENT DeleteBrowse (((ShortName, VersionID) | DataSetID | GranuleUR), InternalFileName*, BrowseCollectionId?,
BrowseGranuleId?)>
<!ATTLIST DeleteBrowse
         SilentDeleteErrorHandling NMTOKEN "0"
>
<!ELEMENT ShortName (#PCDATA)>
<!ELEMENT VersionID (#PCDATA)>
<!ELEMENT DataSetID (#PCDATA)>
<!ELEMENT GranuleUR (#PCDATA)>
<!ELEMENT InternalFileName (#PCDATA)>
<!ELEMENT BrowseCollectionId (#PCDATA)>
<!ELEMENT BrowseGranuleId (#PCDATA)>
<!ELEMENT BrowseCrossReference (((ShortName, VersionID) | DataSetID | GranuleUR), BrowseCollectionId?, BrowseGranuleId?,
InsertTime?, LastUpdate?, DeleteTime?, InternalFileName, BrowseDescription?, BrowseSize)>
<!ELEMENT InsertTime (#PCDATA)>
<!ELEMENT LastUpdate (#PCDATA)>
<!ELEMENT DeleteTime (#PCDATA)>
<!ELEMENT BrowseDescription (#PCDATA)>
<!ELEMENT BrowseSize (#PCDATA)>
```

Element definitions used in Browse ingest insert/update/delete:

DTDVersion

The DTD version of the ingest dtd. In this case, it is BMGTBrowseMetadata.   For example, for the 7.0.3 external test system, this value is '7.0.3'. Note: Currently, this field is ignored by ingest.

DataCenterId

The DataCenterID of the data center from which the ingest is originating.   Note: this field is ignored by ingest.

GranuleUR

The GranuleUR for the granule associated with the browse file.

StartDate

Temporal start date of the data. Currently ignored by ingest.

EndDate

```
Temporal end date of the data. Currently ignored by ingest.
BrowseCrossReference
The tag needed for inserting or updating browse files
DeleteBrowse
The tag needed for deleting browse files
InternalFileName
This is the name of the browse image you will be associating with the granule.
InsertTime
Insert time that the granule was inserted into ECHO.
LastUpdate
The LocalLastUpdate time for the granule in ECHO. The new LastUpdate date must be more recent than old Lastupdate date when
updating browse.
DeleteTime
The time to delete the browse from the system. If DeleteTime tag not used, then deletion occurs instantaneously.
BrowseSize
The size of the browse file being inserted or updated for a granule. The value in the browse ingest file MUST match the
actual file size.
SilentDeleteErrorHandling
=0: will return error message if any needed item (e.g. ShortName, VersionID, DatasetID, or GranuleUR) does not exist.
=1:   turns off error messages
```

7.2.18.2 Insert Browse Example

The file testbrowse.xml (see Figure 7) is an XML example of inserting browse references in to multiple granules. In the example XML we are inserting one sample browse image file into a single granule. This sample will insert a unique browse image file into each granule.

```
<?xml version="1.0" encoding="UTF-8"?>
<BrowseReferenceFile>
 <DTDVersion>1.0</DTDVersion>
 <DataCenterId>EDC</DataCenterId>
 <TemporalCoverage>
   <StartDate>2005-05-12T00:00:00.000Z</StartDate>
   <EndDate>2005-06-12T00:00:00.000Z</EndDate>
 </TemporalCoverage>
 <BrowseCrossReference>
   <GranuleUR>10x10ReferenceGranule0:0</GranuleUR>
   <LastUpdate>2005-05-12 07:13:57.05</LastUpdate>
   <InternalFileName>test.hdf0</InternalFileName>
   <BrowseSize>10</BrowseSize>
 </BrowseCrossReference>
 <BrowseCrossReference>
   <GranuleUR>10x10ReferenceGranule0:1</GranuleUR>
   <LastUpdate>2005-05-12 06:47:12.383</LastUpdate>
   <InternalFileName>test.hdf1</InternalFileName>
   <BrowseSize>11</BrowseSize>
 </BrowseCrossReference>
 <BrowseCrossReference>
   <GranuleUR>10x10ReferenceGranule0:2</GranuleUR>
   <LastUpdate>2005-05-12 06:30:03.576</LastUpdate>
   <InternalFileName>test.hdf2</InternalFileName>
   <BrowseSize>12</BrowseSize>
 </BrowseCrossReference>
 <BrowseCrossReference>
   <GranuleUR>10x10ReferenceGranule0:3</GranuleUR>
```

```
    <LastUpdate>2005-05-12 06:22:19.323</LastUpdate>
    <InternalFileName>test.hdf3</InternalFileName>
    <BrowseSize>13</BrowseSize>
  </BrowseCrossReference>
  <BrowseCrossReference>
    <GranuleUR>10x10ReferenceGranule0:4</GranuleUR>
    <LastUpdate>2005-05-12 05:23:35.37</LastUpdate>
    <InternalFileName>test.hdf4</InternalFileName>
    <BrowseSize>14</BrowseSize>
  </BrowseCrossReference>
</BrowseReferenceFile>
```

Figure 13. Insert browse example: testbrowse.xml

Edit the test sample file to use local granuleURs.

Copy the file testbrowse.xml to the /provider/data/browse ftp location for your provider.

Copy the corresponding browse image files to the /provider/data/browse ftp location for you provider.

From /ingestversion/proc, run ingest.sh.

7.2.18.3 Update Browse Example

The file testbrowseu.xml (see Figure 8) is an XML example of updating browse references in multiple granules. In the example XML we are updating the browse references we inserted from testbrowse.xml for the 5 granules.

```
<?xml version="1.0" encoding="UTF-8"?>
<BrowseReferenceFile>
  <DTDVersion>1.0</DTDVersion>
  <DataCenterId>EDC</DataCenterId>
  <TemporalCoverage>
    <StartDate>2005-05-12T00:00:00.000Z</StartDate>
    <EndDate>2005-06-12T00:00:00.000Z</EndDate>
  </TemporalCoverage>
  <BrowseCrossReference>
    <GranuleUR>10x10ReferenceGranule0:0</GranuleUR>
    <LastUpdate>2005-05-16 07:13:57.05</LastUpdate>
    <InternalFileName>test.hdf1</InternalFileName>
    <BrowseSize>11</BrowseSize>
  </BrowseCrossReference>
  <BrowseCrossReference>
    <GranuleUR>10x10ReferenceGranule0:1</GranuleUR>
    <LastUpdate>2005-05-16 06:47:12.383</LastUpdate>
    <InternalFileName>test.hdf2</InternalFileName>
    <BrowseSize>12</BrowseSize>
  </BrowseCrossReference>
  <BrowseCrossReference>
    <GranuleUR>10x10ReferenceGranule0:2</GranuleUR>
    <LastUpdate>2005-05-16 06:30:03.576</LastUpdate>
    <InternalFileName>test.hdf3</InternalFileName>
    <BrowseSize>13</BrowseSize>
  </BrowseCrossReference>
  <BrowseCrossReference>
    <GranuleUR>10x10ReferenceGranule0:3</GranuleUR>
    <LastUpdate>2005-05-16 06:22:19.323</LastUpdate>
    <InternalFileName>test.hdf4</InternalFileName>
    <BrowseSize>14</BrowseSize>
  </BrowseCrossReference>
  <BrowseCrossReference>
    <GranuleUR>10x10ReferenceGranule0:4</GranuleUR>
    <LastUpdate>2005-05-16 05:23:35.37</LastUpdate>
    <InternalFileName>test.hdf0</InternalFileName>
    <BrowseSize>10</BrowseSize>
  </BrowseCrossReference>
</BrowseReferenceFile>
```

Figure 14. Update browse example: testbrowseu.xml

NOTE: In order for the updates to work, the filename must be the same as the insert filename*. Example: If you inserted a browse image into a granule, and your ingest file was named testbrowse.xml, then your update file must also be named testbrowse.xml.* ***If the file names are not the same, the browse function will treat the ingest file as an insert instead of an update.***

Edit the test sample file to use local granuleURs.

Copy the file testbrowseu.xml to the /provider/data/browse ftp location for your provider.

Copy the corresponding browse image files to the /provider/data/browse ftp location for you provider.

From /ingestversion/proc, run ingest.sh.

7.2.18.4 Delete Browse Example

The file testbrowsed.xml (see Figure 9) is an XML example of deleting browse references. In the example XML, we are removing the browse references we have inserted and updated from testbrowse.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<BrowseReferenceFile>
  <DTDVersion>1.0</DTDVersion>
  <DataCenterId>EDC</DataCenterId>
  <TemporalCoverage>
    <StartDate>2005-04-11T00:00:00.000Z</StartDate>
    <EndDate>2005-04-12T00:00:00.000Z</EndDate>
  </TemporalCoverage>
  <DeleteBrowse SilentDeleteErrorHandling="0">
    <GranuleUR>10x10ReferenceGranule0:0</GranuleUR>
    <InternalFileName>test.hdf1</InternalFileName>
  </DeleteBrowse>
  <DeleteBrowse SilentDeleteErrorHandling="0">
    <GranuleUR>10x10ReferenceGranule0:1</GranuleUR>
    <InternalFileName>test.hdf2</InternalFileName>
  </DeleteBrowse>
  <DeleteBrowse SilentDeleteErrorHandling="0">
    <GranuleUR>10x10ReferenceGranule0:2</GranuleUR>
    <InternalFileName>test.hdf3</InternalFileName>
  </DeleteBrowse>
  <DeleteBrowse SilentDeleteErrorHandling="0">
```

```
      <GranuleUR>10x10ReferenceGranule0:3</GranuleUR>
      <InternalFileName>test.hdf4</InternalFileName>
    </DeleteBrowse>
    <DeleteBrowse SilentDeleteErrorHandling="0">
      <GranuleUR>10x10ReferenceGranule0:4</GranuleUR>
      <InternalFileName>test.hdf0</InternalFileName>
    </DeleteBrowse>
  </BrowseReferenceFile>
```

Figure 15. Delete browse example: testbrowsed.xml

Edit the test sample file to use local granuleURs.

Copy the file testbrowsed.xml to the /provider/data/browse ftp location for your provider.

Copy the corresponding browse image files to the /provider/data/browse ftp location for you provider.

From /ingestversion/proc, run ingest.sh.

7.2.19 Sample Messages

Code Example 219. Remove all online data URLs for a granule, but do not update the last update for granule Gr1 and Gr2.

```
<ProviderAccountService>
  <UpdateMetadata>
    <Granule>
      <Target>
        <ID>Gr1</ID>
        <ProviderLastUpdateDateTime>
          2002-06-03 23:00:00
        </ProviderLastUpdateDateTime>
        <SaveDateTimeFlag><DONTSAVE/></SaveDateTimeFlag>
      </Target>
      <Target>
        <ID>Gr2</ID>
        <ProviderLastUpdateDateTime>
          2002-06-03 23:00:00
        </ProviderLastUpdateDateTime>
        <SaveDateTimeFlag><DONTSAVE/></SaveDateTimeFlag>
      </Target>
      <Delete>
        <QualifiedTag>OnlineAccessURLs</QualifiedTag>
      </Delete>
    </Granule>
  </UpdateMetadata>
</ProviderAccountService>
```

Code Example 220. Deleting a specific online data URL.

```
<ProviderAccountService>
  <UpdateMetadata>
    <Granule>
```

```
      <Target>
        <ID>Gr1</ID>
        <ProviderLastUpdateDateTime>
          2002-06-03 23:00:00
        </ProviderLastUpdateDateTime>
        <SaveDateTimeFlag><DONTSAVE/></SaveDateTimeFlag>
      </Target>
      <Delete>
        <QualifiedTag>
          OnlineAccessURLs/OnlineAccessURL[URL="http://jp.provider.com/Gr1URL"]
        </QualifiedTag>
      </Delete>
    </Granule>
  </UpdateMetadata>
</ProviderAccountService>
```

Code Example 221. Insert a new online resource.

```
<ProviderAccountService>
<UpdateMetadata>
<Granule>
<Target>
  <ID>Gr1</ID>
  <ProviderLastUpdateDateTime>
    2002-06-03 23:00:00
  </ProviderLastUpdateDateTime>
  <SaveDateTimeFlag><DONTSAVE/></SaveDateTimeFlag>
</Target>
<Add>
   <QualifiedTag>GranuleOnlineResources/OnlineResource/OnlineResourceURL </QualifiedTag>
   <MetadataValue>http://jp.provider.com/Gr1URL.metadata</MetadataValue>
</Add>
<Add>
  <QualifiedTag>

GranuleOnlineResources/OnlineResource[OnlineResourceURL="http://jp.provider.com/Gr1URL.metadata"]/OnlineResourceDescription
  </QualifiedTag>
  <MetadataValue>Metadata cashed in Japan</MetadataValue>
</Add>
<Add>
  <QualifiedTag>
   GranuleOnlineResources/OnlineResource[OnlineResourceURL="http://jp.provider.com/Gr1URL.metadata"]/OnlineResourceType
  </QualifiedTag>
  <MetadataValue>Metadata</MetadataValue>
</Add>

<Add>
  <QualifiedTag>
GranuleOnlineResources/OnlineResource[OnlineResourceURL="http://jp.provider.com/Gr1URL.metadata"]/OnlineResourceMimeType
  </QualifiedTag>
  <MetadataValue>text</MetadataValue>
</Add>
</Granule>
```

```
</UpdateMetadata>
</ProviderAccountService>
```

Code Example 222. Update online data.

```
<ProviderAccountService>
<UpdateMetadata>
<Granule>
<Target>
  <ID>Gr1</ID>
  <ProviderLastUpdateDateTime>
    2002-06-03 23:00:00
  </ProviderLastUpdateDateTime>
  <SaveDateTimeFlag><DONTSAVE/></SaveDateTimeFlag>
</Target>
<Update>
  <QualifiedTag>
   OnlineAccessURL[URL="http://jp.provider.com/Gr1URL"]/URL
  </QualifiedTag>
  <MetadataValue>http://jp.provider.com/GRANULES/Gr1URL</MetadataValue>
</Update>
</Granule>
</UpdateMetadata>
</ProviderAccountService>
```

Code Example 223. QA flag update.

```
<ProviderAccountService>
<UpdateMetadata>
<Granule>
<Target>
  <ID>Gr1</ID>
  <ProviderLastUpdateDateTime>
    2002-06-03 23:00:00
  </ProviderLastUpdateDateTime>
  <SaveDateTimeFlag><SAVE/></SaveDateTimeFlag>
</Target>
<Update>
  <QualifiedTag> MeasuredParameter/MeasuredParameterContainer [ParameterName="Param1"]/QAFlags/AutomaticQualityFlag
  </QualifiedTag>
  <MetadataValue>automatic quality flag</MetadataValue>
</Update>
<Update>
  <QualifiedTag> MeasuredParameter/MeasuredParameterContainer
[ParameterName="Param1"]/QAFlags/AutomaticQualityFlagExplanation
  </QualifiedTag>
  <MetadataValue>automatic quality flag explanation</MetadataValue>
</Update>
</Granule>
</UpdateMetadata>
</ProviderAccountService>
```

7.2.20 Error Messages

ECHO processes ingested metadata and sends an ingest summary report via e-mail to the data provider upon a successful ingest. The ingest summary report is presented in XML. This report describes any abnormalities discovered in the input file. In the current version of the ingest process, the input data error handling is applied by input file pre-processing and data ingesting/updating.

7.2.20.1 Input File Validation Error Messages

Before the provider's data is ingested, the input file is analyzed and validated against its applied DTD. If an error is detected, the input file is rejected and an input file error report will be sent to the provider via a pre-registered email address. Below is an example of the input file error report:

Error Message 23. Example of the Input File Error Report.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ECHO Ingest Input File Error Report: Wed Mar 19 10:20:01 EST 2003 -->
<!-- created  by ECHO Ingest System -->
<InputFileErrorSummary>
   <DataProvider>GCC</DataProvider>
   <ProcessStartTime>03/19/2003 10:20:01</ProcessStartTime>
   <CollectionInputFileWithError>
      <XMLValidationError>
         <FileName>VTCCLSR72001101659031903.XML</FileName>
         <ErrorMessage>
         file read took 0.005 seconds

error parsing: Error: Content model for CollectionAssociation does not allow it to end here
 in unnamed entity at line 245 char 26 of [unknown]
         </ErrorMessage>
      </XMLValidationError>
      <NoneXMLFile>VTCCLSR72001101659031903.XML.met</NoneXMLFile>
   </CollectionInputFileWithError>
   <ProcessStopTime>03/19/2003 10:20:02</ProcessStopTime>
</InputFileErrorSummary>
```

7.2.20.1.1 Non-XML File

If the input file does not start with the standard XML file declaration line such as <?xml …>, then this input file is considered as a "non-XML" file. A non-XML file is rejected for ingest process. The error will be reported in the input file error summary report and the error message listed as below:

<NonXMLFile>VTCCLSR72001101659031903.XML.met</NonXMLFile>

7.2.20.1.2 XML File Put in Wrong Entry Directory

For each provider, a directory under their FTP home directory is assigned for putting specific types of input XML files. For instance, all collection metadata XML files might go to a …/collection directory, while all granule XML files might go to a …/granule directory. If the XML file that is deposited in the …/collection directory is not a collection XML file, then this file will be rejected for ingest processing. The error will be reported in the input file error summary report and the error message listed as:

<None…XMLFile>VTCGMOLT200110920011240101.XML</None…XMLFile>

where the … is filled with specific category including "Collection", "Granule", or "Browse" depending on which category of data is examined.

7.2.20.1.3 Invalid XML File

Each input XML file will be validated against its corresponding DTD. If for any reason the DTD validation would fail, this file will be rejected for ingest processing. The error will be reported in the input file error summary report and the error message listed as below:

Error Message 24. XML Validation Error

```
<XMLValidationError>
   <FileName>VTCCLSR72001101659031903.XML</FileName>
   <ErrorMessage>
      file read took 0.005 seconds
error parsing:
Error: Content model for CollectionAssociation does not allow it to end here in unnamed entity at line 245 char 26 of [unknown]
   </ErrorMessage>
</XMLValidationError>
```

The text in the TAG of ErrorMessage is the error message generated by the XML validation utility. It usually stops validation as soon as the first error is detected.

7.2.20.2 Data Ingest Error Messages

When a provider's data has been successfully ingested, an ingest summary report will be sent to the provider via a pre-registered email address. Below is an example of the ingest summary report:

Error Message 25. Example of the Ingest Summary Report.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ECHO Ingest Summary Report: Wed Nov 13 13:55:01 EST 2002 -->
<!-- created  by ECHO Ingest System -->
<IngestSummary>
   <DataProvider>ETE_TEST</DataProvider>
   <IngestStartTime>11/13/2002 13:55:01</IngestStartTime>
   <Collection>
      <InputFiles>
         <File name="VTCCLSR7200211130011240555.XML" size="8424" />
      </InputFiles>
      <Processed total="1">
         <Replacement total="1">
            <DataSets>
               <DataSet shortname="L7CPF" version="2" />
            </DataSets>
         </Replacement>
      </Processed>
   </Collection>
   <Granule>
      <InputFiles>
         <File name="VTCGLSR72002111300112401010425555.XML" size="1967" />
      </InputFiles>
      <Processed total="2">
         <Insertion total="1" />
         <Replacement total="1" />
      </Processed>
   </Granule>
   <IngestStopTime>11/13/2002 13:56:26</IngestStopTime>
</IngestSummary>
```

Five groups of ingest information are included: <Collection>, <Granule>, <Browse>, <Valids>, and <updates>. For browse and valids, only the input files' name and size are listed to show what input files ECHO has processed. For metadata updates, collections and granules, more detailed information is given. The detailed ingest process information is listed under the <Processed> section of the summary report. Under the <Processed> Element, various tags could go under each indicating a transaction or an error. The total number of collections or granules that fall into the transaction or error tag is listed as the tag's attribute. For a collection, a list of dataset short names and version numbers will be listed for each tag that indicates the transaction or error. ECHO breaks a large input file into small chunks to process at one time. Thus, the total in a <Process> tag will not be bigger then 2000. The <Process> Element may be repeated multiple times for an input file.

The following list gives the possible error messages associated with each transaction within the Ingest Summary Report. Some error messages are more complex then others and so will be treated in more detail in the following subsections.

Note: A collection or granule will not be ingested if it produces any of the errors listed below.

7.2.20.2.1 Out of Date Update for Collection or Granule

Error Message 26. Error returned because provider's last update of collection or granule is older then the information recorded in the ECHO database.

```
<Processed total="n">
   ….
    <OutOfDate total="m">
       <DataSets>
          <DataSet shortname="collection's short name" version="version number" />
             ……
       </DataSets>
     </OutOfDate>
        ……
</Processed>
```

Where n, and m is the total occurrences for this round of ingest. If this is for granule then only total number of occurrences will be listed.

7.2.20.2.2 Duplicated Input

Error Message 27. Error returned due to more then one input collection or granule has the same identifier in a single round of ingest.

```
<Processed total="n">
   ….
    <Duplicated total="m">
       <DataSets>
          <DataSet shortname="collection's short name" version="version number" />
             ……
       </DataSets>
     </Duplicated>
        ……
</Processed>
```

Where n, and m is the total occurrences for this round of ingest. If this is for granule then only total number of occurrences will be listed.

This error returned because in the same round of ingest, there are more then one input collection with the same identifier. ECHO reports the incident and takes the first one in the file bearing the newest collection as the one to be ingested and ignores every other one.

7.2.20.2.3 Invalid Spatial Data

7.2.20.2.4

Error Message 28. Error Message 5: Error returned when the spatial coverage area data for a collection or granule is invalid

```
<Processed total="n">
   ….
    <InvalidSpatial total="m">
       <DataSets>
          <DataSet shortname="collection's short name" version="version number" />
             ……
       </DataSets>
     </InvalidSpatial>
        ……
</Processed>
```

Where n, and m are the total occurrences for this round of ingest. If this is for granule then only total number of occurrences will be listed.

This error returned when the spatial coverage area data for the collection is invalid.

7.2.20.2.5 Bad Data

Error Message 29. Error returned at data loading when any piece of data associated with a collection or granule has violated a data constraint (i.e., a character string was used when a number type attribute was specified)

```
<Processed total="n">
   ….
    <RecordWithBadData total="m">
       <DataSets>
          <DataSet shortname="collection's short name" version="version number" />
             ……
       </DataSets>
     </RecordWithBadData>
        ……
</Processed>
```

Where n, and m is the total occurrences for this round of ingest. If this is for granule then only total number of occurrences will be listed.

This error is returned at data loading when any piece of data associated with a collection has violated a data constraint; such as if a character string was sent for a number type attribute. This collection will be ignored and reported back to the data provider.

However, if the data that violated the data constraint is part of the basic information of the collection, then this collection will not make it into the database. If you noticed the total number of collections processed is less then what you submitted, you should contact the ECHO Operations team for an investigation. The detailed information should be in the Operations ingest log file.

7.2.20.2.6 Invalid Deletion

Error Message 30. Error returned when a collection or granule identifier sent for deletion does not exist in the ECHO database

```
<Processed total="n">
```

```
  ….
   <InvalidDeletion total=”m”>
      <DataSets>
         <DataSet shortname=”collection’s short name” version=”version number” />
            …..
      </DataSets>
   </InvalidDeletion>
        …..
</Processed>
```

Where n, and m are the total occurrences for this round of ingest. If this is for granule then only the total number of occurrences will be listed.

7.2.20.2.7 Granule Associated with a Collection that Does Not Exist in the ECHO Database

Error Message 31. Error returned because provider's collection referenced by the granule(s) is either not in the ECHO database or failed to be ingested into the ECHO database.

```
<Processed total="n">
  ..
     <AssociatedCollectionNotExist total="m">
        ..
</Processed>
```

Where n, and m are the total occurrences for this round of ingest.

7.2.20.2.8 Metadata Update Errors

Error Message 32. Not XML file format

```
<MetadataUpdateInputFileWithError>
         <NoneXMLFile>file name</NoneXMLFile>
</MetadataUpdateInputFileWithError>
```

Error Message 33. Not MetadataUpdate XML file

```
<MetadataUpdateInputFileWithError>
         <NoneMetadataUpdateXMLFile>file name</ NoneMetadataUpdateXMLFile>
</MetadataUpdateInputFileWithError>
```

Error Message 34. Invalid metadata update

```
<MetadataUpdateInputFileWithError>
         <XMLValidationError>
                  <FileName>metadataupdate.xml</FileName>
                  <ErrorMessage>xxx</ErrorMessage>
</XMLValidationError>
</MetadataUpdateInputFileWithError>
```

ErrorMessage can be any of the following:

'Metadata update tags do not exist or are incorrect'

'The granule does not exist'

'The new ProviderLastUpdateDateTime is earlier than the current ProviderLastUpdateDateTime in ECHO'

'Attempting to update with no MetadataValue provided'

'Attempting to delete or update non-exist data URL'

'Attempting to insert existing data URL'

'Attempting to delete or update non-exist online resource URL'

'Attempting to insert existing online resource URL'

'QA Flag allows update only'

'Attempting to update QA Flag with incorrect parameter name for'

'Missing online resource type during online resource URL insert'

7.2.20.3 Errors Ignored by ECHO

Wrong Date Format. When the information associated with date is not expressed in a format that is agreed upon between a data provider and ECHO, the date information is ignored. The collection or granule will be ingested in the database with partially incorrect data (missing dates).

For instance, a data provider notified ECHO that all their date information is going to be expressed in the format of "yyyy-mm-dd hh24:mi:ss" (such as "2002-03-15 14:50:00"), but the date information in the input file looks like

<LastUpdate>2002-3-15 2:50:00PM</LastUpdate>

then this date will be ignored. ECHO stores this collection or granule with empty data for the last update date.

7.3 Data Management Service

The Data Management Service uses groups defined in the Group Management Service to grant permissions to certain users. This allows the actual membership of a group to be managed (optionally) by someone other than the data provider. The Data Management Service allows a provider to deny access to everyone at one time to describe which granules and collections are affected. Then, in a subsequent transaction, permission is granted to a group or set of groups to access either all or particular granules or collections (also according to the rules).

The actual implementation of the Data Management Service centers on rules and conditions. Conditions exist independently of rules, allowing for reusability. For example, a RollingTemporalCondition of 30 days can be used in a restriction rule to prevent access to young metadata. The same RollingTemporalCondition can then be applied to an internal test group to permit a provider to verify the correctness of said metadata. If the provider decides to change the young metadata threshold from 30 days to 45 days, only one condition must be changed (because the restriction and permission reference the same RollingTemporalCondition).

The scope of the current implementation is in terms of an access control list being applied to collections and granules as defined by the provider of the metadata. Collections can be protected such that neither the collection can be seen nor any of its constituent granules. Collections can also be protected so the collection metadata is visible, but only certain granules are protected from view. Individual granules can be hidden.

The act of hiding granules or collections does not affect queries in the system. When the results of the query are presented, the access control lists are checked at the time of the present transaction. This means that if the access control list is updated while a user is working with a result set, the user will be able to see the things that they had been previously restricted from viewing. If an item that is hidden is presented, then an indication that it is hidden is returned instead of the actual metadata.

The Data Management Service allows the hiding of individual granules for searching and/or for ordering. It is possible to hide granules in a collection according to a rule, so the provider does not need to hide each granule individually. There are three basic rules:

Temporal: Granules that are produced during a certain time period are restricted.

Floating Temporal: Granules that are produced within the last N days are restricted.

Restriction Flag Based: Granules that have certain restriction flag values are restricted.

After a condition is created, a Comparator is applied to it to form a rule. Examples of Comparators include <, >, <=, =, <>, etc. Comparators and conditions form the basis for a rule's evaluation. The above mentioned 30 day RollingTemporalCondition is meaningless without the < Comparator. Similarly, a provider may create a RollingTemporalCondition of 10 years and use the > Comparator to prevent access to extremely old metadata.

The RollingTemporalCondition and TemporalCondition rules must also indicate which of three granule timestamps to apply:

Production date/time: the date and time at which the data was produced.

Insert date/time: the date and time that the metadata record for a collection or granule was inserted into the provider's metadata repository.

Acquisition date/time: the date and time that the original observation occurred.

These timestamps are referred to as the Temporal Type, an enumerated type with options PRODUCTION, INSERT, and ACQUISITION. For production and insert, standard rules apply for >=, >, <=, and <. For example, if you specify a rule using >= and the temporal type PRODUCTION, ECHO will select granules where the production_date_time timestamp is greater than or equal to the specified date.

Temporal type ACQUISITION is used in a slightly different manner. A granule's acquisition timestamp is actually represented as a range specified by a beginning acquisition date/time and an ending acquisition date/time. If you specify >= and the temporal type ACQUISITION, ECHO will select all granules where the specified date/time was greater than or equal to the beginning date/time of the granule. Choosing <= selects granules with a date/time that is equal to or earlier than the ending acquisition date/time.

Rules also accept an ActionType. Examples of ActionType include order, view, browse, etc. The ActionType allows a provider to restrict access to metadata based upon what action the user is taking. A provider may restrict the ordering of some piece of metadata, but not the browsing or discovery of the metadata.

Lastly, if the rule created is a permission, a group is associated with the rule. This allows the provider to allow access to metadata to a particular subset of the ECHO community. The group referenced by a rule does not have to include the provider as a manager. The provider creating the permission may reference any group they wish.

Rules are evaluated optimistically. That is, if a particular user tries to take an action upon a piece of metadata, permissions supersede restrictions. If the metadata is restricted, but the user belongs to a group that has permission on the metadata, access is granted.

7.3.1 Transactions

7.3.1.1 ListConditions

This transaction lists the conditions the current user has created. The response contains the ConditionDescriptor, which identifies the condition, and indicates its type (temporal, rolling temporal, restriction flag, or Boolean).

7.3.1.2 CreateBooleanCondition

Boolean conditions are the most straightforward of the three because they simply wrap a true/false flag.

Code Example 224. Boolean Condition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
        <CreateBooleanConditionRequest>
                <BooleanCondition>
                        <ConditionName>true condition</ConditionName>
                        <description>my true data condition</description>
                        <BooleanFlag>true</BooleanFlag>
                </BooleanCondition>
        </CreateBooleanConditionRequest>
```

```
</DataManagementService>
```

The The BooleanFlag set to 'true' indicates that this is a true condition. A false condition (BooleanFlag set to 'false') is not allowed in ECHO. The value of the Boolean condition is evaluated at the rule level using a Comparator. The only valid comparator is EQUALS.

7.3.1.3 PresentBooleanCondition

This transaction presents the specified Boolean condition. If no names are specified, all the user's Boolean conditions are presented. The response contains a name and description of each condition, in addition to a flag that identifies the condition as Boolean.

7.3.1.4 CreateRestrictionFlagCondition

This transaction creates a named restriction flag condition. The RestrictionFlagCondition is made up of the condition name, description and lower value (which is an integer threshold) for each condition. The response is an indication of whether the action succeeded.

Code Example 225. RestrictionFlag Condition

```
<DataManagementService>
        <CreateRestrictionFlagConditionRequest>
           <RestrictionFlagCondition>
               <ConditionName>Restriction Flag Example</ConditionName>
               <description>Restriction Flag Condition Example</description>
               <Lower>8</Lower>
           </RestrictionFlagCondition>
        </CreateRestrictionFlagConditionRequest>
</DataManagementService>
```

7.3.1.5 PresentRestrictionFlagCondition

This transaction presents a restriction flag condition, as described above. If no names are specified, all of the user's restriction flag conditions are presented.

7.3.1.6 CreateRollingTemporalCondition

Rolling temporal conditions are similar to temporal conditions in that they are both time based, however, rolling temporal conditions make use of a duration to infer a start and stop date. A provider wishing to restrict metadata produced within the previous 30 days either must modify the stop date of a temporal condition every day or use a Rolling Temporal Condition with a duration of 30 days. The rolling temporal condition is the simpler solution.

Code Example 226. Creating a rolling temporal condition with a duration of 30 days for insert date

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
   <CreateRollingTemporalConditionRequest>
      <RollingTemporalCondition>
         <ConditionName>
            young data condition
         </ConditionName>
         <description>
            young data condition
         </description>
         <Duration>
            <TemporalUnit>
               <DAYS/>
            </TemporalUnit>
            <TemporalDuration>
               30
            </TemporalDuration>
         </Duration>
         <TemporalType>
            <INSERT/>
         </TemporalType>
      </RollingTemporalCondition>
   </CreateRollingTemporalConditionRequest>
</DataManagementService>
```

You must provide a name and description for the rolling temporal condition. In addition to that information, you must also provide a temporal unit, temporal duration, and temporal type (production, acquisition, or insert – to identify the granule timestamp to use in applying the condition). These fields allow you to specify (up to the hour) what duration you are interested in controlling. The API gives you the flexibility to create a young data condition (30 Days) as well as an old data condition (10 years).

After conditions are created, rules can be enforced that reference the conditions. Each rule uses a Comparator to determine how to evaluate the embedded condition. In the Boolean and temporal condition examples, a Comparator of == makes sense (a not equals in the temporal condition would also make sense and would control all metadata NOT in the time frame specified). In the rolling temporal condition example, a Comparator of < makes sense. A > Comparator would also make sense if the rolling temporal condition had a duration of 10 years and you wished to control access to old metadata.

7.3.1.7 PresentRollingTemporalCondition

This transaction presents a rolling temporal condition, as described above. If no names are specified, all of the user's rolling temporal conditions are presented.

7.3.1.8 CreateTemporalCondition

Temporal conditions cover a specific time and are identifiable by their distinct start and stop dates. Temporal conditions are used to identify granules whose date of acquisition exists within a specific time frame. Application of a temporal condition may occur if there is a known window where a sensor on your satellite was turned off for maintenance and you wish to restrict granules during that time period.

Code Example 227. Creating a temporal condition for insert date.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
   <CreateTemporalConditionRequest>
      <TemporalCondition>
         <ConditionName>
            month of september 2001
         </ConditionName>
         <description>
            The door on the satellite was closed during september 2001.  I wish to
            block access to the metadata produced during that time period (because
            I know it to be bad).
         </description>
         <StartTime>2001-09-01</StartTime>
         <StopTime>2001-09-30</StopTime>

         <TemporalType>
            <INSERT/>
         </TemporalType>
      </TemporalCondition>
   </CreateTemporalConditionRequest>
</DataManagementService>
```

When you create a temporal condition you must specify a name and a description, a start and stop date during which the temporal condition is active, and a temporal type (production, acquisition, or insert).

7.3.1.9 PresentTemporalCondition

This transaction presents a named temporal condition. If no names are specified, all temporal conditions will be presented. The response includes the following information:

- ConditionName
- Description
- StartTime
- StopTime
- TemporalType (production, acquisition, or insert)

7.3.1.10 DeleteCondition

This transaction attempts to remove the specified condition. If the condition is actively enforced as a rule, the condition will not be deleted. The request must include a condition descriptor for each condition to be deleted, which identifies the condition and the condition type (temporal, rolling temporal, restriction flag, or Boolean).

7.3.1.11 EnforceRule

This transaction assigns a particular data rule to the provider's data. This transaction could be used to grant visibility to a particular group for all collections for the provider. It could also be used to restrict visibility for particular collections from a provider.

The request message includes the data rule, which is comprised of:

- Rule name and description.
- Rule type (restriction or permission).
- Action type (view, order, or browse).
- Condition name.
- Condition type (temporal, rolling temporal, restriction flag, or Boolean).

Note: Since condition names are unique for a provider, the condition type is a redundant field when submitting this data type to a transaction. However, the field is useful in responses from ECHO, because it is used to specify the exact type of Condition the rule is using. Without it, a user would be unable to present the condition.

- Comparator (less than, less than or equal, greater than, greater than or equal, equals, or not equals).
- Group name. The current implementation ignores the group name if the rule type is a restriction.

Note: The difference between a restriction and the permission is that a restriction is defined upon a user while permission is defined upon a group. A rule may be applied to multiple groups.

- Data holding (the collection to be restricted or allowed).

Note: A rule may be applied to multiple collections.

7.3.1.12 PresentRule

This transaction returns the details of a named data rule, as described above.

7.3.1.13 ListRules

This transaction generates a list of the rules the current user has assigned to their metadata holdings. The response includes a list of the current user's rules.

7.3.1.14 CancelRule

This transaction is not yet implemented.

7.3.1.15 DeleteRule

This transaction deletes a named data rule. The response indicates whether the action succeeded.

7.3.2 *Error Messages*

All errors caused by invoking transactions within the Data Management Service result in a rollback condition. That is, if you pass N rules to the EnforceRule transaction, and ECHO fails to create the Nth rule, no rules are created. This behavior is consistent with the rest of the ECHO system. The DataManagementService is only accessible to providers. Science users and guests cannot invoke any transactions in this service.

This table gives the possible error messages associated with each transaction within the Data Management Service.

Table 19: DMS error messages.

Transaction	Error Message	Description
CreateBooleanCondition	<The missing item> is not valid.	The user specified a blank name or description
	Condition <name> already exists in your profile.	The user specified a name that is already in use.
CreateRestrictionFlagCondition	<The missing item> is not valid.	The user specified a blank name or description
	Condition <name> already exists in your profile.	The user specified a name that is already in use.
PresentRestrictionFlagCondition	RestrictionFlag condition '<name>' does not exist	User specified a name that does not exist
	RestrictionFlag condition 'null ' does not exist	User specified a blank name
CreateRollingTemporalCondition	<The missing item> is not valid.	The user specified a blank name or description
	Condition <name> already exists in your profile.	The user specified a name that is already in use.
	Invalid or null parameters were used.	The user specified a blank or non-numeric Day or Year.
	Stop date is before start date.	The user specified a stop date before the start date.

Transaction	Error Message	Description
DeleteCondition	Unable to delete condition. Condition ‘<name>’ not found.	The user specified a condition type and condition name that does not exist.
	Unable to delete condition. Condition ‘null’ not found.	The user specified a blank condition.
	Unable to delete condition. Condition ‘<name>’ is actively being used in a rule.	The user specified a condition referenced by a rule.
DeleteRule	Unable to find rule: <name>.	The user specified a rule that does not exist.
	Unable to find rule: null.	The user specified a blank rule name.
EnforceRule	Unable to create rule: <the missing item> is not valid.	The user specified a blank rule name or description.
	Unable to create rule: The name ‘<name>’ is already in use.	The user specified a rule name that already exists.
	Unable to create rule: the condition referenced does not exist.	The user specified a condition that does not exist.
	Data Value ‘<value>’ is not valid.	The user specified a collection or granule that does not exist.
	Unable to create rule: the group specified does not exist.	The user specified a RuleType of PERMISSION but did not provide a GroupName.
PresentBooleanCondition	Boolean condition ‘<name>’ does not exist.	The user specified a name that does not exist.
	Boolean condition ‘null’ does not exist.	The user specified a blank name.
PresentRollingTemporalCondition	Rolling temporal condition ‘<name>’ does not exist.	The user specified a name that does not exist.
	Rolling temporal condition ‘null’ does not exist.	The user specified a blank name.
PresentTemporalCondition	Temporal condition ‘<name>’ does not exist.	The user specified a name that does not exist.
	Temporal condition ‘null’ does not exist.	The user specified a blank name.
PresentRule	Rule ‘<name>’ does not exist.	The user specified a name that does not exist.
	Rule ‘null’ does not exist.	The user specified a blank name.

7.3.3 *Restrictions and Permissions: Examples*

Restrictions in ECHO apply to all users (including Guests). Their counterpart (permissions) apply to group(s) (created through the GroupManagementService) and hence affect specific users of ECHO (Guests are not included). Permissions (as their name implies) grant access to metadata that is restricted. Creating permissions is the same as creating restrictions except a Group is required when enforcing the rule.

7.3.3.1 Creating a Restriction using a Boolean Condition

Below is an example of how a provider would restrict access to multiple collections using a single Boolean condition access restriction rule. After this transaction is invoked, presentation of the collection level metadata and any granule-level metadata associated with the collection will be restricted.

Code Example 228. Restricting access to a specific collection with a Boolean condition.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
   <EnforceRuleRequest>
      <DataRule>
         <RuleName>
            MODIS Level 1A V001 Restriction
         </RuleName>
         <Description>
            prevents access to the MODIS Level 1A V001 collection as well as
            all granules associated with the collection.
         </Description>
         <RuleType>
            <RESTRICTION/>
         </RuleType>
         <ActionType>
            <VIEW/>
         </ActionType>
         <ConditionType>
            <BOOLEAN/>
         </ConditionType>
         <ConditionName>
            true condition
         </ConditionName>
         <Comparator>
            <EQUALS/>
         </Comparator>
         <DataHolding>
            <DataType>
               <COLLECTION/>
            </DataType>
            <DataValue>
               MODIS Level 1A V001
            </DataValue>
            <DataValue>
               MODIS Level 2A V001
            </DataValue>
         </DataHolding>
      </DataRule>
   </EnforceRuleRequest>
</DataManagementService>
```

The above example restricts viewing access on the MODIS Level 1A V001 and MODIS Level 2A V001 collections using the Boolean condition named 'true'. Because the Comparator specified is EQUALS and the Boolean condition's BooleanFlag is set to 'true', access to the collection is restricted. If a provider wished to prevent viewing and ordering of the collection, they would have to create two separate restrictions and specify an ActionType of VIEW in the first restriction, and an ActionType of ORDER in the second restriction.

7.3.3.2 Create Permissions for multiple groups using Boolean condition

This example illustrates how to give permission to multiple groups to access restricted metadata. After this transaction is invoked, presentation of the collection level metadata and any granule-level metadata associated with the collection will be allowed for the selected groups.

Code Example 229. Creating permissions for multiple groups through boolean conditions.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
        <EnforceRuleRequest>
                <DataRule>
                        <RuleName>
                                QA Permission
                        </RuleName>
                        <Description>
                                Allows the MODIS QA group to view all data
                        </Description>
                        <RuleType>
                                <PERMISSION/>
                        </RuleType>
                        <ActionType>
                                <VIEW/>
                        </ActionType>
                        <ConditionType>
                                <BOOLEAN/>
                        </ConditionType>
                        <ConditionName>
                                true condition
                        </ConditionName>
                        <Comparator>
                                <EQUALS/>
                        </Comparator>
                        <GroupName>
                                MODIS QA
                        </GroupName>
                        <GroupName>
                                MODIS PIs
                        </GroupName>
                        <DataHolding>
                                <DataType>
                                             <COLLECTION/>
                                </DataType>
                                <DataValue>
                                        ALL
                                </DataValue>
                        </DataHolding>
                </DataRule>
        </EnforceRuleRequest>
</DataManagementService>
```

7.4 Orders and Options

7.4.1 Triggers and Communication Protocols

Currently, there are three types of user actions on orders that trigger ECHO to communicate with the providers and require a response from the providers. These three types of user actions are:

- **SUBMIT:** This action is taken when the user is requesting to process an order. Currently, all providers must support this transaction.
- **QUOTE:** This action is taken when the user is requesting to get a quote from a provider for an order before the order is submitted.
- **CANCEL:** This action is taken when the user is requesting to cancel an order that is already submitted and in processing.

There are two types of protocol that ECHO communicates with the providers. The first type is solely used to communicate with an ECS-like provider. The communication uses ODL (Object Description Language) formatted messages over an open socket. The second type is to communicate using SOAP (Simple Object Access Protocol) which uses a proprietary ECHO format that is expressed in XML. These two types of communication protocols are usually referred to as either ODL (also referred to as ECS) or SOAP.

The provider must specify a URI (Uniform Resource Identifier), sometimes known as network address, in order for ECHO to communicate with the provider. The network address is usually either an IP or HTTP address. For instance, most ODL transactions point to a network location similar to the following: 64.48.91.99, while most SOAP transactions point to a network location similar to the following: http://organization.domain/soap/servlet/rpcrouter. The exact URI location should be made available by the provider for each of the user actions that the provider supports. This location is where ECHO ultimately sends the actual transaction message.

7.4.2 Provider Response

Upon receiving any of the three order actions – submit, quote, or cancel – ECHO constructs a message with the applicable document format based on the incoming transaction and sends out the message to the specified provider location using the applicable communication protocol.

The provider must supply complete information including a ProviderTrackingID, a StatusMessage, and additional order information for transactions like submit and quote, back to ECHO.

- **ProviderTrackingID:** A unique identifier for a provider order. ECHO provides a tracking ID on its first communication to the provider. The provider has the option of replacing it with its own tracking ID. ECHO will use the “current” tracking ID for all subsequent communications with the provider.
- **StatusMessage:** Timestamped message records the status history of the provider order that can be viewed by the user using PresentProviderOrder or PresentOrder transaction in OrderEntryService.
- **Additional information for submit response:** Order information including total price for the order (mandatory), price for data, price for the media that stores the data, price for shipping, price for handling, any discount made to the total price, quantity of media used to store the data, estimated ship date for the order, latest data for cancellation of the order, and additional free text information.
- **Additional information for quote response:** Quote information including total price for the order (mandatory), price for data, price for the media that stores the data, price for shipping, price for handling, any discount made to the total price, recommended media to store the data, quantity of media used to store the data, expiration date for the quote (mandatory), and additional free text information.

7.4.2.1 Synchronous Action and Asynchronous Action

The provider is required to respond to the ECHO message immediately either synchronously or asynchronously. In the synchronous response, the provider sends back an answer (either accept or reject) and includes all required information such as TrackingID, StatusMessage, and additional transaction specific information in the response. In an asynchronous response, the provider is allowed to respond with just an acknowledgement message that includes a TrackingID and a StatusMessage. To ultimately complete the request, the provider must send back its decision and any required information later via the transactions in the Provider Order Management Service.

7.4.2.2 Retry Mechanism

If no response is received from the provider, ECHO will issue another request after waiting for a period of time. The provider can set a policy for the maximum numbers of attempts for retry and waiting time between retries using the ProviderAccountService API.

7.4.3 Options

7.4.3.1 Introduction to Options

The ECHO application frameworks include a mechanism that allows properties to be dynamically defined and attached to entities within the system. A property is a named value, similar to an attribute. The "properties" mechanism has been created to enable easy, code-free extensibility of ECHO entities. Order and User Profile objects currently rely upon this properties mechanism to enable portions of their state to be dynamically defined and set.

The properties mechanism is useful for managing properties that are defined and used by systems that are outside of the context of ECHO. A data provider that ECHO interacts with is a good example of such an external system. The data provider may require that specific information accompany any order or request for data. Furthermore, portions of that information may be uniquely specific to requests for data sent to that particular provider. This is a situation where applying the property mechanism makes sense. Here, the provider-specific information will be defined and stored abstractly as a collection of properties that must be set by the user during the order entry process.

The ECHO API uses the parameters Option and OptionSelection to communicate property definitions and property values respectively. An Option parameter defines a property value that may be set through the ECHO API. An OptionSelection parameter is used to set the value that is defined by the option or to communicate the value that is currently set for the option. The diagram in Figure 16 shows the structure and relationships of the option parameters, as defined in the XML API.

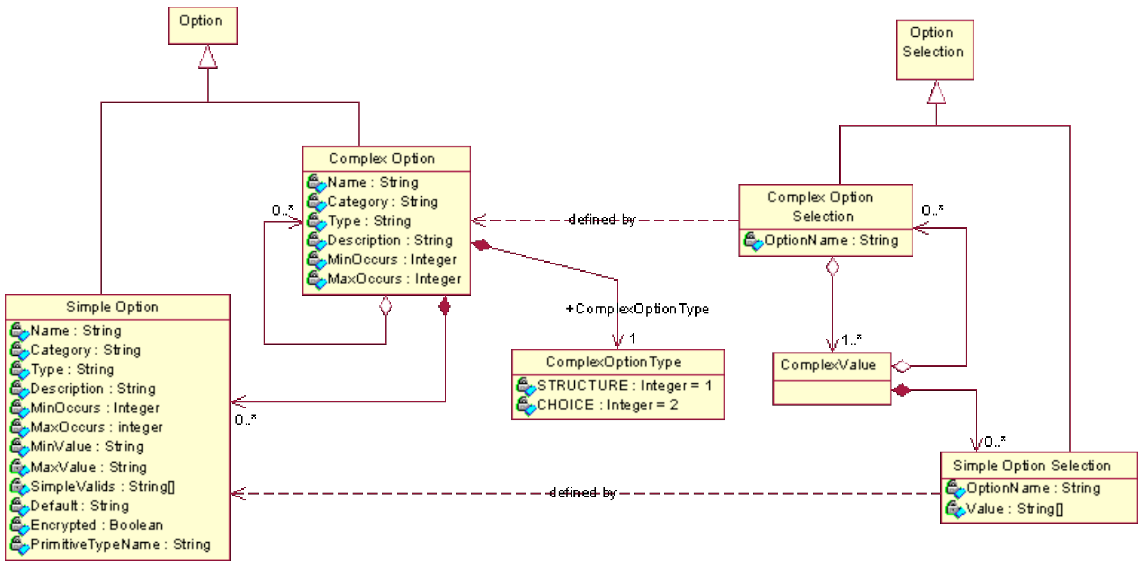


Figure 16. Structure and relationships of the Option Parameters.

There are two types of options: simple and complex. A simple option describes an atomic piece of information of a primitive type. The "Type" field indicates the primitive value type that is expected. A simple option may also specify that multiple instances of that type may be input--similar to an array. The "MinOccurs" and "MaxOccurs" field values define the multiplicity of a corresponding value, while the MinValue and MaxValue fields define the range of allowable values. A simple option may declare the complete set of distinct valid values that may be set for a simple option selection of that type. These are defined in the "Valids" field. Finally, a default value may be specified in the "Default" field.

The second type of option is ComplexOption. A complex option is an option that defines a grouping of input parameters, sub-options, and/or choices. Like simple options, complex options may declare the allowable multiplicity of values. These are defined in the "MinOccurs" and "MaxOccurs" fields. The complex option "Type" field indicates the validation behavior that will be applied to the corresponding complex value instance. There are two types of complex value: structure and choice. A “choice” type indicates that for each complex option selection value (remember that there may be many values), a sub-value for only one of the sub-options contained by the complex option may be set. A “structure” type, on the other hand, allows a complex value to be set that contains set sub-values for each sub-option defined by the complex option.

An OptionSelection is a structure that represents the values that correspond to an Option. Mirroring the option hierarchy, there are two types of option selection: SimpleOptionSelection, and ComplexOptionSelection. A simple option selection has a name that matches the name of its simple option, and contains one or more primitive values. A complex option selection also has a name that matches the name of its [complex] option, and contains one or more complex values. A complex value is a container of option selections. It will contain zero or more simple options and complex options, as defined by the complex option and its associated “Type.”

Due to the reflexive nature of options and option selections, reading and generating appropriate OptionSelections can be confusing. An example definition with some explanations and mapping between the definition and the selection has been provided in Appendix B.

7.4.3.2 Order Options

When a provider registers a collection into ECHO, one step in the process is to identify what additional information is needed when an order is placed. This information typically falls into the following categories:

Delivery Specification – Electronic or Media

Associated Supporting Files – Inclusion in the delivery package of additional related files

File Packaging Instructions – How to group the files into a package (zip archive, tar ball, etc.)

Processing Service Options – Identification of parameters needed to invoke additional processing before delivery occurs. Common services include temporal, spectral and spatial subsetting and the generation of higher-level products with the same spatial and temporal footprint as the original granule.

This information is made available to clients who then gather the corresponding settings from the end user. These settings are included in the OrderOptions field. There will be a corresponding setting for each definition provided.

The Option Selection is a hierarchical data structure and is defined in the options.xsd file. This file is available from the Reference section of the ECHO Web site at http://eos.nasa.gov/echo. The leaf nodes of the hierarchy must be Simple Options. Non-leaf nodes are Complex Options and are used to express how to group its “contained” options. The root node can be either a Complex Option or a Simple Option. The Order Options is a list of Option Selections, one for each Option Definition defined by the Provider. The name of the Option Selection corresponds to the name of the Option Definition for which this setting applies. The hierarchy of Option Selections should also parallel the Option Definition hierarchy in that each node should be of the same type (simple or complex) and should have been validated by the Option Definition template structure. In the case of Complex Options Selections, the contained options’ names will reflect the user’s selections. For instance, if the Complex Option is of type CHOICE, then there will be exactly one contained option, and its name will indicate which the user chose. If the type is STRUCTURE, then potentially all of them could be present. The minOccurs attribute in the definition allows for optional children if it is set to 0.

It is up to the provider to interpret this hierarchy and to construct the information they specified was needed in the first place when the definition was identified. This can be accomplished by the adapter if needed.

```
<complexType name="OptionSelection">
 <choice>
  <element name="ComplexOptionSelection"
       type="tns:ComplexOptionSelection"/>
  <element name="SimpleOptionSelection" type="tns:SimpleOptionSelection"/>
 </choice>
</complexType>

<complexType name="ComplexOptionSelection">
 <sequence>
  <element name="Name" type="string"/>
  <element name="ComplexValues"
       type="tns:ComplexValue" maxOccurs="unbounded"/>
 </sequence>
</complexType>

<complexType name="ComplexValue">
 <sequence>
  <element name="SimpleOptionSelections"
       type="tns:SimpleOptionSelection"
       minOccurs="0" maxOccurs="unbounded"/>
  <element name="ComplexOptionSelections"
       type="tns:ComplexOptionSelection"
       minOccurs="0" maxOccurs="unbounded"/>
 </sequence>
</complexType>

<complexType name="SimpleOptionSelection">
 <sequence>
  <element name="Name" type="string"/>
  <element name="Values" type="string" maxOccurs="unbounded"/>
 </sequence>
</complexType>
```

7.4.4 Order Composition

An ECHO Order request is composed of all the information a provider should need in order to place an order. This includes the following:

an order identifier (orderId) that is used to uniquely identify an order

a list of line items (lineItems) that describes the actual items to be ordered and their options

information about who is placing, receiving, and paying for the order (userInfo)

the identifier associated with the client who facilitated placing the order (clientIdentity)

These fields are present in both the quote and submit request messages, which are described in the interface.wsdl file. The fields are defined in the types.xsd file and are described in the next sections in more detail. Both files are available in the Reference section of the ECHO web site.

```
<part name="orderId" type="order:OrderId"/>
<part name="lineItems" type="order:ListOfLineItems"/>
<part name="userInfo" type="order:UserInformation"/>
<part name="clientIdentity" type="xsd:string"/>
```

7.4.4.1 Order Identifier

The OrderId provides identifying information about an order. The ProviderId is ECHO's name for the provider. The OrderId is ECHO's unique identifier for the order. The TrackingId is where the Provider's unique order tracking number is maintained for future reference. DataCenterId is the name by which the provider expects to be called when receiving an order, which could be different than the ProviderId.

(Note: When ECHO was first conceived, some providers had two systems. For instance, GSFC had both an ECS and a V0 system. To distinguish, ECHO called them GSFC-ECS and GSFC-V0. However, the order message expected the name GSFC. So, the DataCenterId field is used so that order messages could be filled in appropriately.)

```
<complexType name="OrderId">
  <sequence>
    <element name="ProviderId" type="string"/>
    <element name="OrderId" type="string"/>
    <element name="TrackingId" type="string"/>
    <element name="DataCenterId" type="string"/>
  </sequence>
</complexType>
```

7.4.4.2 Line Items

The Line Item represents the item to be ordered (collection or granule) and all supporting information describing exactly what is being ordered. The first field of a Line Item is the GranuleUr. This is the granule universal reference that is defined as the provider's unique identifier for the granule. This is not the ECHO unique identifier because it is assumed that the provider's order system does not know anything about ECHO. It is optional in the case where the Line Item being ordered is a data set. The next field is the DataSetId. This is the provider's unique identifier for a collection. It will be filled in with the identifier of the collection to which the granule belongs if the order is for a granule, and the identifier of the collection being ordered if the order is for a collection. The Quantity field allows for an order to specify how many of an item should be ordered. If the order is for electronic delivery (i.e. FTP), then the quantity should always be one. If the order is for the delivery of pre-packaged media, then the quantity can be considered the number of media that are being requested. The AuthenticationKey field is set based on the client calling SetAuthenticationKey. (As of Version 6, this field can be no longer than 16 characters.) Finally, the options specified by the user regarding this line item are passed in as OrderOptions.

```
<complexType name="ListOfLineItems">
 <sequence>
   <element name="lineItems" type="tns:LineItem" maxOccurs="unbounded"/>
 </sequence>
</complexType>
```

```
<complexType name="LineItem">
 <sequence>
   <element name="GranuleUr" type="string" minOccurs="0"/>
   <element name="DataSetId" type="string"/>
   <element name="Quantity" type="integer"/>
   <element name="AuthenticationKey" type="string" minOccurs="0"/>
   <element name="OrderOptions" type="options:OptionSelection"
         minOccurs="0" maxOccurs="unbounded"/>
 </sequence>
</complexType>
```

7.4.4.3 User Information

User Information is used to convey to the Provider who is placing the order. It consists of the ECHO User ID and the associated addresses for the order. While only the Contact Address is required, it should be noted that some providers require all three addresses to be completed in order to fulfill the order.

```
<complexType name="UserInformation">
 <sequence>
   <element name="EchoUserId" type="string"/>
   <element name="ShippingAddress"
         type="tns:ShippingAddress" minOccurs="0"/>
   <element name="BillingAddress"
         type="tns:BillingAddress" minOccurs="0"/>
   <element name="ContactAddress"
         type="tns:ContactAddress"/>
 </sequence>
</complexType>
```

7.4.4.3.1 Shipping Address

The Shipping Address is used to indicate where an order should be physically shipped. If an order is for electronic delivery only, then this field is superfluous. The Name field contains the first and last name of the person to whom the order should be shipped. The Business Name contains the name of the company to whom the order should be shipped. This is an optional field. The Contact Name is the name of the person to be contacted if there are any questions about shipping. The Phone Number contains a string with the telephone number of the contact name in case there are shipping issues. The Special Instruction field allows for more information for the delivery (e.g. “Leave the package by the Banyan tree.”) Finally, the Address field is the destination of the delivery.

```
<complexType name="ShippingAddress">
  <sequence>
    <element name="Name" type="tns:Name"/>
    <element name="BusinessName" type="string" minOccurs="0"/>
    <element name="ContactName" type="string"/>
    <element name="PhoneNumber" type="string"/>
    <element name="SpecialInstruction" type="string"/>
    <element name="EmailAddress" type="string"/>
    <element name="Address" type="tns:Address"/>
  </sequence>
</complexType>
```

7.4.4.3.2 Billing Address

The Billing Address is similar to the Shipping Address, and identifies who will be paying the bill for the order. There is no Special Instruction field for this case.

```
<complexType name="BillingAddress">
  <sequence>
    <element name="Name" type="tns:Name"/>
    <element name="BusinessName" type="string" minOccurs="0"/>
    <element name="ContactName" type="string"/>
    <element name="PhoneNumber" type="string"/>
    <element name="EmailAddress" type="string"/>
    <element name="Address" type="tns:Address"/>
  </sequence>
</complexType>
```

7.4.4.3.3 Contact Address

The Contact Address is also similar, but it identifies who is placing the order.

```
<complexType name="ContactAddress">
  <sequence>
    <element name="Name" type="tns:Name"/>
    <element name="PhoneNumber" type="string"/>
    <element name="EmailAddress" type="string"/>
    <element name="BusinessName" type="string"/>
    <element name="Address" type="tns:Address"/>
  </sequence>
</complexType>
```

7.4.4.3.4 Name

This structure is simply a way to distinguish the first and last name of a person. First Name can be augmented with middle initials or a middle name if needed. Last Name can be augmented with generational modifies (e.g. Jr.) if needed.

```
<complexType name="Name">
  <sequence>
    <element name="FirstName" type="string"/>
    <element name="LastName" type="string"/>
  </sequence>
</complexType>
```

7.4.4.3.5 Address

The address field is a structure that describes an address. The ID field is a name that the user uses to describe the address (e.g. Home, Work, Project1, etc.). The US Format flag is used to indicate whether a provider can rely on the address looking like a standard US address. The rule is that the system will validate those addresses that have the US Format flag set by checking that Street1, City, State, Zip Code, and Country are all set. If the US Format flag is clear, then only Street1 and Country are required. ECHO allows for up to 5 street address lines in either case.

```
<complexType name="Address">
  <sequence>
    <element name="Id" type="string"/>
    <element name="USFormat" type="boolean"/>
    <element name="Street1" type="string"/>
    <element name="Street2" type="string" minOccurs="0"/>
    <element name="Street3" type="string" minOccurs="0"/>
    <element name="Street4" type="string" minOccurs="0"/>
    <element name="Street5" type="string" minOccurs="0"/>
    <element name="City" type="string" minOccurs="0"/>
    <element name="State" type="political:State" minOccurs="0"/>
    <element name="ZipCode" type="string" minOccurs="0"/>
    <element name="Country" type="political:Country"/>
  </sequence>
</complexType>
```

7.4.4.4 Client Identity

The client identity is unique for each ECHO client. Orders created by an ECHO client carry this unique id. Providers can use the client identity to determine the source of any given order. This information is sent along with the other order information to the provider via the provider proxy.

The client identity takes the form of a string generated by the client and is expected to look like “Client Name v1.0.”

Note: ECS truncates client identities at 16 characters. SOAP users have no such limitations.

This applies to the Submit, Quote, and Cancel transactions.

7.5 Provider Order Management Service

7.5.1 Transactions

7.5.1.1 AcceptProviderOrderSubmission

This transaction is used when the user has submitted an order, and the provider decides to accept this order. The provider provides SubmissionInformation about this order, which includes total price, shipping/handling, latest cancel date, estimated ship date and a StatusMessage that is a short message about this order to the user. In addition, the provider is able to specify the UpdateMechanism. If the provider chooses to set this value to be "MANUAL", this means the provider is willing to update the status manually through other transactions in the Provider Order Management Service until the order is closed. In this case, the provider order state changes from SUBMITTING to PROCESSING. If the provider sets UpdateMechanism to "NONE", the provider is not going to update the order status. In this case, the provider order state changes from SUBMITTING to CLOSED.

Note: The date format for latest cancel date and estimated ship date must follow "mm/dd/yyyy".

Code Example 230. AcceptProviderOrderSubmission transaction (updating status manually).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <AcceptProviderOrderSubmissionRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <SubmissionInformation>
         <totalPrice>10.00</totalPrice>
         <latestCancelDate>11/10/02</latestCancelDate>
         <estimatedShipDate>11/20/02</estimatedShipDate>
         <dataPrice>5.00</dataPrice>
         <mediaPrice>1.00</mediaPrice>
         <shipping>3.00</shipping>
         <handling>1.00</handling>
         <discount>0</discount>
         <quantityOfMedia>1</quantityOfMedia>
         <additionalInformation>office will close on 11/18/02</additionalInformation>
      </SubmissionInformation>
      <StatusMessage>This order has been accepted and will be processing in 24 hours.</StatusMessage>
      <UpdateMechanism>
         <MANUAL/>
      </UpdateMechanism>
   </AcceptProviderOrderSubmissionRequest>
</ProviderOrderManagementService>
```

Code Example 231. AcceptProviderOrderSubmission transaction (closing it immediately without guarantee for updating status).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <AcceptProviderOrderSubmissionRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <SubmissionInformation>
         <totalPrice>10.00</totalPrice>
         <estimatedShipDate>11/20/02</estimatedShipDate>
         <dataPrice>5.00</dataPrice>
         <mediaPrice>1.00</mediaPrice>
         <shipping>3.00</shipping>
         <handling>1.00</handling>
         <discount>0</discount>
         <quantityOfMedia>1</quantityOfMedia>
         <additionalInformation>office will close on 11/18/02</additionalInformation>
      </SubmissionInformation>
      <StatusMessage>This order has been accepted and will be shipped upon completion.</StatusMessage>
      <UpdateMechanism>
         <NONE/>
      </UpdateMechanism>
   </AcceptProviderOrderSubmissionRequest>
</ProviderOrderManagementService>
```

7.5.1.2 RejectProviderOrderSubmission

This transaction is used when the user has submitted the order, and the provider decides to reject this order. The provider provides the user a StatusMessage to explain the reason of rejection. In this transaction, the provider order state changes from SUBMITTING to SUBMIT_REJECTED.

Code Example 232. RejectProviderOrderSubmission transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <RejectProviderOrderSubmissionRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <StatusMessage>You are not allowed to order item #111 because … </StatusMessage>
   </RejectProviderOrderSubmissionRequest>
</ProviderOrderManagementService>
```

7.5.1.3 CancelProviderOrder

This transaction is used either when the user has requested to cancel the provider order using the CancelOrder transaction in the User Account Service and the provider decides to accept the request, or when the provider decides to cancel the provider order. In this first case, the provider order state changes from CANCELLING to CANCELLED. In this second case, the provider order state changes from PROCESSING to CANCELLED. In either case, the provider must provide a StatusMessage about this cancellation.

Code Example 233. CancelProviderOrder transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <CancelProviderOrderRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <StatusMessage>This order is cancelled as requested. </StatusMessage>
   </CancelProviderOrderRequest>
</ProviderOrderManagementService>
```

7.5.1.4 RejectProviderOrderCancellation

This transaction is used when the user has requested to cancel the provider order using the CancelOrder transaction in the User Account Service and the provider decides to reject this request. The provider must explain the reason for rejection in the StatusMessage. In this transaction, the provider order state changes from CANCELLING to PROCESSING.

Code Example 234. RejectProviderOrderCancellation Transaction

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <RejectProviderOrderCancellationRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <StatusMessage>The item is ready to ship out, can not cancel anymore. </StatusMessage>
   </RejectProviderOrderCancellationRequest>
</ProviderOrderManagementService>
```

7.5.1.5 CloseProviderOrder

This transaction is used when the provider has finished processing part or all of the order. The provider documents the progress of the order in a StatusMessage. In this transaction, the provider order state changes from PROCESSING to CLOSED.

Code Example 235. CloseProviderOrder transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <CloseProviderOrderRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <StatusMessage>Your order has been shipped out.</StatusMessage>
   </CloseProviderOrderRequest>
</ProviderOrderManagementService>
```

7.5.1.6 SupplyProviderQuote

This transaction is used when the user has requested to quote the provider order using the QuoteOrder transaction in the Order Entry Service and the provider decides to supply the information quoted. The provider supplies a ProviderQuote that includes the total price, shipping/handling, recommended media, quote expiration date etc. and a StatusMessage to the user. In this transaction, the provider order state changes from QUOTING to QUOTED. Note that the date format for quote expiration date must follow “mm/dd/yyyy”.

Code Example 236. SupplyProviderQuote transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <SupplyProviderQuoteRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <ProviderQuote>
         <totalPrice>8.00</totalPrice>
         <quoteExpirationDate>11/30/02</quoteExpirationDate>
         <dataPrice>5.00</dataPrice>
         <mediaPrice>1.00</mediaPrice>
         <shipping>3.00</shipping>
         <handling>1.00</handling>
         <discount>0.20</discount>
         <quantityOfMedia>1</quantityOfMedia>
         <recommendedMedia>CD</recommendedMedia>
         <additionalInformation>office will close on 11/18/02</additionalInformation>
      </ProviderQuote>
      <StatusMessage>quote as requested. </StatusMessage>
   </SupplyProviderQuoteRequest>
</ProviderOrderManagementService>
```

7.5.1.7 RejectProviderQuote.

This transaction is used when the user has requested to quote the provider order using the QuoteOrder transaction in the Order Entry Service and the provider decides to reject the quote request. The provider provides the user a StatusMessage to explain the reason of rejection. In this transaction, the provider order state changes from QUOTING to QUOTE_REJECTED.

Code Example 237. RejectProviderQuote transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <RejectProviderQuoteRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <StatusMessage>Our system does not support quote at this time. </StatusMessage>
   </RejectProviderQuoteRequest>
</ProviderOrderManagementService>
```

7.5.1.8 UpdateStatusMessage.

This transaction allows the provider to update the status message of an order to inform the user. This transaction does not associate with any provider order state change.

7.5.1.9 ChangeTrackingID

The tracking ID is the identity of ECHO order in the provider's system. In some cases, the provider wants to change this ID in their system. This transaction is used to register this change to ECHO system.

7.5.1.10 Provider Order History

Each provider has the capability to present the history of the orders that have been submitted to its data center. The PresentClosedOrder transaction lists all the orders that have been closed and the PresentOpenOrder lists all the orders that have been submitted to the provider but are still in process. Each of these transactions lists detailed information of each order. To list only the order Id and its order state, POMS provides PresentClosedOrderSummary and PresentOpenOrderSummary.

7.5.1.10.1 PresentClosedOrder

This transaction enables a provider to view detailed information about orders that are no longer active. The provider has the choice of either specifying or not specifying the provider order states in the request. If provider order states are specified, the information of orders in the specified state is returned. If no provider order state is specified, then all the persisted orders that are closed are returned.

Code Example 238. Present all closed orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService SYSTEM "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentClosedOrderRequest/>
</ProviderOrderManagementService>
```

Code Example 239. Present closed orders in CLOSED state.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService SYSTEM "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentClosedOrderRequest>
      <ClosedProviderOrderState>
         <CLOSED/>
      </ClosedProviderOrderState>
   </PresentClosedOrderRequest>
</ProviderOrderManagementService>
```

7.5.1.10.2 PresentClosedOrderSummary

This transaction enables a provider to view brief information about the orders that are no longer active. The provider has the choice of either specifying or not specifying the provider order states in the request. If provider order states are specified, the information of orders in the specified state is returned. If no provider order state is specified, then all the persisted orders that are closed are returned.

Code Example 240. Present summary for all closed orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService SYSTEM "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentClosedOrderSummaryRequest/>
</ProviderOrderManagementService>
```

Code Example 241. Present summary for closed orders in CLOSED state.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService SYSTEM "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentClosedOrderSummaryRequest>
      <ClosedProviderOrderState>
         <CLOSED/>
      </ClosedProviderOrderState>
   </PresentClosedOrderSummaryRequest>
</ProviderOrderManagementService>
```

7.5.1.10.3 PresentOpenOrder

This transaction enables a provider to view detailed information about the orders that are still open. The provider has the choice of either specifying or not specifying the provider order states in the request. If provider order states are specified, the information of orders in the specified state is returned. If no provider order state is specified, then all the persisted orders that are still open are returned.

Code Example 242. Present all open orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService SYSTEM "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentOpenOrderRequest/>
</ProviderOrderManagementService>
```

Code Example 243. Present open orders in PROCESSING state.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService SYSTEM "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentOpenOrderRequest>
      <OpenProviderOrderState>
         <PROCESSING/>
      </OpenProviderOrderState>
   </PresentOpenOrderRequest>
</ProviderOrderManagementService>
```

7.5.1.10.4 PresentOpenOrderSummary

This transaction enables a provider to view brief information about the orders that are still open. The provider has the choice of either specifying or not specifying the provider order states in the request. If provider order states are specified, the information of orders in the specified state is returned. If no provider order state is specified, then all the persisted orders that are still open are returned.

Code Example 244. Present summary for all open orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService SYSTEM "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentOpenOrderSummaryRequest/>
</ProviderOrderManagementService>
```

Code Example 245. Present summary for open orders in PROCESSING state.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService SYSTEM "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentOpenOrderSummaryRequest>
      <OpenProviderOrderState>
         <PROCESSING/>
      </OpenProviderOrderState>
   </PresentOpenOrderSummaryRequest>
</ProviderOrderManagementService>
```

7.5.1.11 PresentProviderOrder

This transaction is used to present the complete specification and description of a provider order. It is useful when a provider wants to check the status of one or more individual orders.

Code Example 246. PresentProviderOrder transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService SYSTEM "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
"http://api.echo.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
    <PresentProviderOrderRequest>
        <OrderID>000</OrderID>
    </PresentProviderOrderRequest>
</ProviderOrderManagementService>
```

7.5.2 Provider Order Management Service Error Messages

This table gives the possible error messages associated with each transaction within the Provider Order Management Service.

Table 20: POMS error messages.

Transaction	Error Message	Description
AcceptProviderOrderSubmission RejectProviderOrderSubmission CancelProviderOrder RejectProviderOrderCancellation SupplyProviderQuote RejectProviderQuote CloseProviderOrder UpdateStatusMessage ChangeTrackingID PresentOpenOrder PresentOpenOrderSummary PresentClosedOrder PresentClosedOrderSummary	[SessionManager] The service {ProviderOrderManagementService} that you attempted to access either does not exist or you do not have the appropriate access level to use it.	This error is returned when trying to access this transaction by a user who does not have the privilege for this transaction.
	Order <OrderID> does not exist in the system	This error is returned when trying to access an order using the order ID that does not exist for this user.
	Invalid provider tracking Id.	This error is returned when trying to access an order using the tracking ID that does not exist for this provider.
	Invalid state for ProviderOrder.	This error is returned when trying to update order from one state to the other state that does not conform to the pre-defined order state flow (e.g., A provider try to set the order state to be “RejectSubmission” on an order at the state of “Closed”).
	[SessionManager] The service {ProviderOrderManagementService} that you attempted to access either does not exist or you do not have the appropriate access level to use it.	This error is returned when trying to access this transaction by a user who does not have the privilege for this transaction.

8 Service Partner Interface

8.1 Extended Service Management

The Extended Service Management [service] allows ECHO partners to introduce and maintain extended service interfaces and implementations. An extended service is a Web service that is offered to the community by an ECHO partner. The service must be compatible with ECHO's goals and may use or manipulate information that can be found using the ECHO system.

A Service Interface describes a Web service and defines the messages and parameters necessary for using it. A Service Interface is described by a WSDL document that contains the types, import, message, portType, and binding elements. It is an abstract definition of a Web service, and is used to describe a specific type of service. Service Interfaces may be defined and maintained by ECHO partners, but are hosted and managed within the ECHO system. Centrally managing interface definitions is intended to encourage the standardization and reuse of service interfaces within the ECHO community. The transaction for registering a Service Interface is defined in the RegisterServiceInterface subsection below.

A service implementation is a Web service that is hosted by an ECHO partner and provides the operations that are defined by a particular service definition (as specified in a Service Interface). In ECHO 6.0, these implementations are referred to as "Services". Like a Service Interface, the Service definition is also captured within a WSDL document. This WSDL Service description document will contain import and service elements (as defined by the WSDL specification). At least one of the import elements will contain a reference to the WSDL service interface document, using the address of the document hosed within the ECHO system. A Service WSDL document can contain references to more than one Service Interface WSDL document. The import element in a WSDL service implementation document contains two attributes. The namespace attribute value is a URL that matches the targetNamespace in the service interface document (as defined by the ECHO system). The location attribute is a URL that is used to reference the WSDL document that contains the complete service interface definition. The Service WSDL definition is hosted by the service provider. Finally, the binding attribute on the port element refers to the specific access point, which will respond to SOAP messages of the specified Service. The transaction for registering a service implementation is defined in the RegisterService subsection.

This separation of the definition of a service into its interface definition and implementation information, both as separation WSDL files, follows contemporary best practices for leveraging WSDL. For more information about this best practice, refer to the whitepaper written by IBM & Microsoft, http://www.uddi.org/pubs/wsdlbestpractices-V1.07-Open-20020521.pdf.

8.1.1 Transactions

8.1.1.1 RegisterServiceInterface

This transaction is used to introduce a new service interface into the system for approval. Once the interface is submitted, it will be reviewed and approved (or rejected) by the ECHO operations group. Once the interface is approved, the submitting partner will receive an e-mail notification communicating that the interface definition has been adopted for use by the ECHO community. The approval e-mail message will include the definition of the unique name adopted for the interface within the ECHO system, as well as the location of the ECHO-hosted WSDL document, which must be referenced by any participating service implementation.

Code Example 247. RegisterServiceInterface transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ExtendedServiceManagement PUBLIC "-//ECHO ExtendedServiceManagement (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ExtendedServiceManagement.dtd">
<ExtendedServiceManagement>
  <RegisterWebServiceRequest>
    <NetworkAddress>
      http://localhost:7001/echo/testwsdl/TestAmazonWebServicesInstance.wsdl
    </NetworkAddress>
    <TaxonomyCategory>
      <TaxonomyName>TEST_TAXONOMY</TaxonomyName>
      <NodeKeyValue>marsans@gst.com</NodeKeyValue>
    </TaxonomyCategory>
  </RegisterWebServiceRequest>
</ExtendedServiceManagement>
```

8.1.1.2 RegisterWebServiceInterface

This transaction is used to introduce a new service interface into the system for approval. Once the interface is submitted, it will be reviewed and approved (or rejected) by the ECHO operations group. Once the interface is approved, the submitting partner will receive an e-mail notification communicating that the interface definition has been adopted for use by the ECHO community. The approval e-mail message will include the definition of the unique name adopted for the interface within the ECHO system, as well as the location of the ECHO-hosted WSDL document, which must be referenced by any participating service implementation.

Code Example 248. RegisterWebServiceInterface transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ExtendedServiceManagement PUBLIC "-//ECHO ExtendedServiceManagement (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ExtendedServiceManagement.dtd">
<ExtendedServiceManagement>
  <RegisterWebServiceInterfaceRequest>
    <NetworkAddress>
       http://localhost:7001/echo/testwsdl/TestAmazonSearch-interface.wsdl
    </NetworkAddress>
    <ServiceDescription>
       This is the a service interface for testing use.  It represents the tmodel interface for a testOrganization.
    </ServiceDescription>
    <TaxonomyCategory>
      <TaxonomyName>TEST_TAXONOMY</TaxonomyName>
      <NodeKeyValue>arsans@gst.com</NodeKeyValue>
    </TaxonomyCategory>
  </RegisterWebServiceInterfaceRequest>
</ExtendedServiceManagement>
```

8.1.1.3 RegisterAdvertisement

This transaction is used to add a new advertisement service into the system for approval. Once the advertisment is submitted, it will be reviewed and approved (or rejected) by the ECHO operations group. Once the advertisement is approved, the submitting partner will receive an e-mail notification communicating that the advertisement has been adopted for use by the ECHO community.

Code Example 249. RegisterAdvertisement transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ExtendedServiceManagement PUBLIC "-//ECHO ExtendedServiceManagement (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ExtendedServiceManagement.dtd">
<ExtendedServiceManagement>
  <RegisterAdvertisementRequest>
    <ServiceName>
```

```
      SampleAdvertisment
    </ServiceName>
    <AccessPoint>
      http://localhost:7001/someplace/AdvertisemnetPage.html
    </AccessPoint>
    <ServiceDescription>
      This is an advertisement service used for testing.
    </ServiceDescription>
    <TaxonomyCategory>
      <TaxonomyName>TEST_TAXONOMY</TaxonomyName>
      <NodeKeyValue>marsans@gst.com</NodeKeyValue>
    </TaxonomyCategory>
  </RegisterAdvertisementRequest>
</ExtendedServiceManagement>
```

8.1.1.4 RegisterWebServiceGUI

This transaction is used to add a new Web service GUI into the system for approval. Once the service GUI is submitted, it will be reviewed and approved (or rejected) by the ECHO operations group. Once the service GUI is approved, the submitting partner will receive an e-mail notification communicating that the service GUI has been adopted for use by the ECHO community.

Code Example 250. RegisterWebServiceGUI transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ExtendedServiceManagement PUBLIC "-//ECHO ExtendedServiceManagement (v6.0)//EN"
"http://api.echo.nasa.gov/echo/dtd/ExtendedServiceManagement.dtd">
<ExtendedServiceManagement>
  <RegisterWebServiceGUIRequest>
    <ServiceName>
      SampleServiceGUI
    </ServiceName>
    <AccessPoint>
      http://localhost:7001/someplace/ServiceGUIPage.html
    </AccessPoint>
    <ServiceDescription>
      This is a Web service GUI used for testing.
    </ServiceDescription>
    <AssociatedWebService>
     0c748a20-2d09-11d9-9977-b8a03c50a862
    </AssociatedWebService>
  </RegisterAdvertisementRequest>
</ExtendedServiceManagement>
```

Appendix A: Acronyms

API	Application Program Interface
DTD	Data Type Definition
ECHO	EOS Clearing House
ECS	EOSDIS Core System
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
ESDIS	Earth Science Data and Information System
FTP	File Transfer Protocol
GCMD	Global Change Master Directory
GIS	Geographic Information System
GML	Geography Markup Language
GMT	Greenwich Mean Time
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ODL	Object Description Language
OGC	OpenGIS Consortium
ORNL	Oak Ridge National Laboratory
PGE	Product Generation Executable
PSA	Product Specific Attribute
QA	Quality Assurance
RPC	Remote Procedure Call
RMI	Remote Method Invocation
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
UDDI	Universal Description, Discovery, and Integration
UR	Universal Reference
URL	Uniform Resource Locator
WSDL	Web Services Definition Language
XML	eXtensible Markup Language
XSLT	eXtensible Style Language Transformation

Appendix B: Annotated Options Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v6.0)//EN" "http://api.echo.nasa.gov/echo/dtd/CatalogService.dtd">
 <OrderEntryService>
 <PresentCatalogItemResponse>
 <BooleanResult>
 <BooleanResultType>
  <REQUEST_SUCCEEDED />
  </BooleanResultType>
  </BooleanResult>
 <CatalogItem>
  <CatalogItemID>5054</CatalogItemID>
 <CatalogEntryType>
  <PRODUCT />
  </CatalogEntryType>
  <ProviderID>1003</ProviderID>
  <productDescription />
  <ProductListPrice>0.0</ProductListPrice>
<!--
Option Definition For Product 5054
Product 5054 can be ordered using one of three production options (Ancillary, Native, and Production History). Also it can
be received via CD-ROM or FTP Push.
-->
 <Option>
 <ComplexOption>
  <Name>ECS-TEST PKG1</Name>
  <OptionCategory>Package Option</OptionCategory>
<!--
The type of this complex option is 'STRUCTURE'. Because it is a structure, each of the sub-options with a MinOccurs
attribute value of one or more must be filled in.
-->
 <ComplexOptionType>
  <STRUCTURE />
  </ComplexOptionType>
  <Description>Default Packaging Options</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
<!—
This simple option defines three valid primitive values: 'Ancillary Granule', 'Native Granule', and 'Production History
Granule'. One of these values must be selected.
-->
 <SimpleOption>
  <Name>Production Option</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>Production Option</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
 <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
   <SimpleValids>
  <ValidPrimitive>Ancillary Granule</ValidPrimitive>
  <ValidPrimitive>Native Granule</ValidPrimitive>
  <ValidPrimitive>Production History Granule
  </ValidPrimitive>
  </SimpleValids>
  </SimpleOption>
 <ComplexOption>
  <Name>Media Type</Name>
  <OptionCategory>Package Option
  </OptionCategory>
<!--
The type of this complex option is 'CHOICE'. Because it is a choice, exactly one of the sub-options defined must be
selected. One of the two media type options, 'CDROM' or 'FTP Push' must be selected.
-->
 <ComplexOptionType>
  <CHOICE />
  </ComplexOptionType>
  <Description>Media Type</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
   <ComplexOption>
  <Name>CDROM</Name>
  <OptionCategory>Package Option</OptionCategory>
 <ComplexOptionType>
  <STRUCTURE />
  </ComplexOptionType>
  <Description>CDROM</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
<!--
This simple option defines three valid String values: 'Gzip', 'Unix',
and 'None'. One of these values must be selected.
-->
 <SimpleOption>
  <Name>Compression</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>Compression Type</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
 <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
   <SimpleValids>
  <ValidPrimitive>GZip</ValidPrimitive>
  <ValidPrimitive>Unix</ValidPrimitive>
  <ValidPrimitive>None</ValidPrimitive>
  </SimpleValids>
  </SimpleOption>
<!--
This simple option defines two valid String values: 'Rockridge', and
'ISO9660'. One of these values must be selected. This option is also
optional, as the value of MinOccurs is zero.
-->
```

```
 <SimpleOption>
  <Name>DDISTMEDIAFMT</Name>
  <OptionCategory>Package option</OptionCategory>
  <Description>Dist Media Format</Description>
  <MinOccurs>0</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
 <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
   <SimpleValids>
   <ValidPrimitive>Rockridge</ValidPrimitive>
  <ValidPrimitive>ISO9660</ValidPrimitive>
  </SimpleValids>
  </SimpleOption>
  </ComplexOption>
 <ComplexOption>
  <Name>FTP Push</Name>
  <OptionCategory>Package Option</OptionCategory>
   <ComplexOptionType>
  <STRUCTURE />
  </ComplexOptionType>
  <Description>FTP Push</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
 <!--
This simple option defines three valid String values: 'Gzip', 'Unix',
and 'None'. One of these values must be selected.
-->
 <SimpleOption>
  <Name>Compression</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>Compression Type</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
   <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
   <SimpleValids>
  <ValidPrimitive>GZip</ValidPrimitive>
  <ValidPrimitive>Unix</ValidPrimitive>
  <ValidPrimitive>None</ValidPrimitive>
  </SimpleValids>
  </SimpleOption>
<!--
This simple option defines a default value: 'Rockridge' The default
value will be filled in as part of the parent structure even if this
option is not set explicitly. Alternately, because this option defines
no valid value list, any string value may be supplied.
-->
 <SimpleOption>
  <Name>DDISTMEDIAFMT</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>Dist Media Format</Description>
  <MinOccurs>0</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
   <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
 <Default>
  <Value>Rockridge</Value>
  </Default>
  </SimpleOption>
<!--
Because this option defines no valid value list, any string value may
Be supplied.
-->
 <SimpleOption>
  <Name>FTPHOST</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>FTP Host</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
 <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
  </SimpleOption>
<!--
This simple option defines a default value: 'anonymous' The default
value will be filled in as part of the parent structure even if this
option is not set explicitly. Alternately, because this option defines
no valid value list, any string value may be supplied. Finally, the
'Encrypted' field value of 'True' indicates that an encryption
mechanism should be used to secure this String value when sending in a
message.
-->
 <SimpleOption>
  <Name>FTPPASSWORD</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>FTP Password</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>True</Encrypted>
   <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
   <Default>
  <Value>anonymous</Value>
  </Default>
  </SimpleOption>
<!--
Because this option defines no valid value list, any string value may be
supplied.
```

```
-->
 <SimpleOption>
  <Name>FTPPUSHDEST</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>Target Directory</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
   <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
  </SimpleOption>
<!--
This simple option defines a default value: 'anonymous' The default
value will be filled in as part of the parent structure even if this
option is not set explicitly. Alternately, because this option defines
no valid value list, any string value may be supplied.
-->
 <SimpleOption>
  <Name>FTPUSER</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>User to log in as</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
 <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
 <Default>
  <Value>anonymous</Value>
  </Default>
  </SimpleOption>
  </ComplexOption>
  </ComplexOption>
  </ComplexOption>
  </Option>
  </CatalogItem>
  </PresentCatalogItemResponse>
  </OrderEntryService>
```